UNIVERSIDAD CARLOS III

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB J2EE PARA LA GESTIÓN DE EVENTOS

PROYECTO DE FINAL DE CARRERA

Leganés, Diciembre 2014

Autor : Sergio Sánchez Blázquez

Tutor : Javier Fernández Muñoz

Título: Diseño e implementación de una aplicación web J2EE para la gestión de eventos

Autor: Sergio Sánchez Blázquez

Director: Javier Fernández Muñoz

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis padres, a quienes debo su trabajo y dedicación para darme una formación profesional.
A Mariano, Jose y Eduardo por sus consejos y motivaciones constantes.
A Paola, por su apoyo incondicional.
A mi tutor, Javier, por haberme guiado.

¡Gracias a todos!

Resumen

En los últimos tiempos, las redes sociales han adquirido una importancia notable y aplicaciones como Tuenti, Twitter o Facebook resultan familiares para cualquier usuario de la red. Con la llegada de la Web 2.0, las redes sociales en Internet ocupan un lugar relevante en el campo de las relaciones personales. Gracias a ellas, podemos interactuar con facilidad con nuestros amigos o conocer gente nueva sin ningún esfuerzo.

En nuestro país, varias de estas redes sociales gozan de gran popularidad. Sin embargo, la mayoría de ellas se basan en la participación colectiva a través de colaborar y compartir con otros usuarios ya conocidos.
En ciertas ocasiones, el usuario desea realizar una actividad y necesita encontrar otros usuarios con sus mismas aficiones.
Existen algunas aplicaciones para llevar a cabo esta labor, pero desafortunadamente no son muy populares en nuestro país.

El proyecto que se detalla a continuación pretende dar una solución concreta a este problema. Se trata de una aplicación para crear y administrar eventos y buscar usuarios a los que les pueda interesar dicho evento. Igualmente se pueden buscar eventos que hayan sido creados con anterioridad. La aplicación ha sido diseñada para ser intuitiva y fácil de usar por el usuario.

A lo largo de las siguientes páginas se dará un repaso a las tecnologías que se han usado así como la evolución histórica de las mismas. También se hará hincapié en el diseño e implementación de la aplicación web.

Abstract

In recent years, social networks have gained significant importance and applications such as Tuenti, Twitter or Facebook are familiar to any network user. With the advent of Web 2.0, social networking sites occupy a prominent place in the field of personal relationships.
Thanks to them, we can interact easily with our friends or meet new people without any effort.

In our country, several of these social networks are popular. However, most of them are based on collective participation through collaboration and sharing with others already known.
Sometimes, the user wants to perform an activity and need to find other users with similar interests.
There are some applications to perform this task, but unfortunately are not very popular in our country.

The project described below is intended to give a concrete solution to this problem. This is an application for creating and managing events and find users who may be interested that event. Also you can search events that have been previously created. The application has been designed to be intuitive and easy to use by the user.

Throughout the following pages will give an overview of the technologies that have been used as well as the historical evolution of the same. It also will focus on the design and implementation of the web application.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1. Introducción

El propósito de este proyecto es la implementación de una aplicación de gestión de eventos, para poder compartir actividades con otros usuarios.También proporcionará otros servicios como la gestión de grupos y usuarios.

Para llevar a cabo la aplicación se usará Spring 3, uno de los frameworks J2EE más populares hoy en día. También se utilizará otro framework muy popular como Hibernate, para desarrollar la capa de persistencia.

La aplicación web, tratará de acercarse al concepto Web 2.0 mediante el uso de tecnologías específicas, como pueden ser AJAX y Google Maps API.

Empezaremos explicando los motivos y características que han gestado esta idea, para en capítulos posteriores explicar el análisis, diseño e implementación del mismo.

1.2. Motivación del proyecto

El mundo de las redes sociales no deja de evolucionar, desarrollándose aplicaciones cada vez más complejas y orientadas la mayoría de ellas a conectar grupos de amigos.

Sin embargo, no siempre todos los amigos comparten las mismas aficiones, por lo que es interesante tener la posibilidad de conocer a gente con esa misma afición en particular.

La intención del proyecto es desarrollar una aplicación de gestión de eventos, donde los usuarios puedan añadir y compartir actividades con personas que no pertenezcan a su grupo más cercano. De este modo, la aplicación pretende ayudar a los usuarios a encontrar a otros con las mismas aficiones.

También la intención es crear una aplicación que sea fácil de utilizar, alejándonos de las cada vez más complejas aplicaciones actuales.

Para ello, se ha decidido usar gran parte de la tecnología de Spring,junto con el framework Hibernate y JPA, y comprobar cómo pueden facilitarnos la creación de una aplicación web, desde su construcción, pasando por su desarrollo y su posterior mantenimiento.

También se le sacará partido al Web 2.0 mediante el uso de AJAX y Google Maps API.

1.3. Objetivos

El objetivo del proyecto es realizar una aplicación que permita a un usuario apuntarse a eventos que son creados por otros usuarios. Del mismo modo, el usuario puede crear eventos que serán mostrados a los demás usuarios.

Otro objetivo principal es desarrollar en la aplicación la manera de gestionar esos usuarios y los grupos que pueden formar entre ellos. Además, la aplicación debe ser sencilla para el usuario y tener una interfaz atractiva e intuitiva.

Por lo tanto, los objetivos principales del proyecto son los siguientes:

- Desarrollo de una aplicación que gestione eventos, grupos y usuarios.
- Tratamiento y almacenamiento de la información para ser accedida en cualquier momento.
- Diseñar una interfaz intuitiva y atractiva para el usuario.

Aparte de los objetivos principales, hay otros objetivos secundarios que se pretenden alcanzar y que tienen que ver con las herramientas que se van a utilizar para el desarrollo. Estos objetivos secundarios son los siguientes:

- Implementar y diseñar la aplicación web usando Spring como principal framework web.
- Utilizar el framework de Hibernate para la persistencia de datos.
- Utilizar las definiciones y reglas de la especificación de JPA, trabajando con anotaciones para el mapeo objeto-relacional.
- Acercarse al concepto de Web 2.0. usando AJAX y la API de Google Maps.

1.4. Contenido

El proyecto se ha estructurado en tres grandes partes bien diferenciadas: una destinada a estudiar el funcionamiento del framework Spring y cómo se integra con las diferentes tecnologías que se han utilizado para desarrollar el proyecto; la segunda trata de cómo se ha diseñado e implementado la aplicación que usará todas estas tecnologías y por último las conclusiones a las cuales se ha llegado, además de información extra aportada mediante anexos.

La primera parte de la memoria comprende de los capítulos 1 al 2. El primer capítulo (Introducción) sirve como ayuda para poder comprobar la motivación de este proyecto, así como los objetivos que se desean alcanzar. En el segundo capítulo (Estado del arte) se trata de explicar el significado del Web 2.0, y así, entender mejor las características que tendrá nuestra aplicación web. Se hablará sobre el uso de J2EE como tecnología de desarrollo. Por último se analizará en detalle el funcionamiento del framework Spring MVC 3, realizando comparativas con otros frameworks del mercado. También hablaremos sobre la integración de otros frameworks a la aplicación como es el caso de Hibernate y la especificación JPA.

La segunda parte de la memoria comprende de los capítulos 3 al 6 y se centra en el análisis, diseño, implementación y pruebas de la aplicación web. En la parte del análisis se mostrarán tanto los requisitos de usuario como los requisitos software.En el capítulo de diseño, se mostrarán los diagramas del proyecto, además de aportar información sobre la arquitectura de la aplicación o las clases del modelo.En el capítulo de implementación se analizará qué pasos se han seguido a la hora de realizar la aplicación.Y por último, en el capítulo de pruebas se indicarán las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.

La parte final de la memoria está compuesta por las conclusiones y resultados que se han obtenido en la realización del estudio e implementación de la aplicación web. También se reflejan las dificultades que han ido apareciendo a lo largo del proyecto y las alternativas de mejora para una posible continuación del proyecto en el futuro. También se incluyen las referencias bibliográficas hechas a lo largo del proyecto junto con los anexos para la correcta utilización y comprensión del sistema. En dichos anexos se explica una lista detallada de los costes que supone implantar la aplicación web, además de realizar una comparación entre los frameworks más usados del mercado y explicar en qué consisten los patrones de diseño utilizados en el proyecto, como son el caso del patrón MVC y el patrón DAO para el acceso a la capa de persistencia de datos.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo, trataremos de explicar las principales tecnologías que se han utilizado dentro del proyecto. Empezaremos comentando la tendencia Web 2.0 y qué tecnologías están desarrolladas dentro de este concepto, haciendo especial hincapié en las que se han utilizado en el proyecto como son las llamadas asíncronas al servidor, para aumentar la interacción con el usuario.

Lo siguiente será comentar la tecnología J2EE para el desarrollo del proyecto, remarcando algunas de sus características principales, que nos servirá de introducción al mundo de los frameworks J2EE.

Una vez visto lo que es un framework y qué puede hacer por nosotros como desarrolladores web, entraré en detalle en el estudio de Spring MVC 3, como principal framework del proyecto, estudiando sus características más importantes. También comentaré la integración de las otras tecnologías que se han utilizado en el proyecto como pueden ser Hibernate y JPA.

2.2. Web 2.0

En agosto de 1995 Netscape -compañía pionera desarrolladora de la que durante los primeros años de vida de Internet fue la aplicación más importante e influyente: el navegador de Internet– salía a bolsa en el NASDAQ estadounidense. Este hito se considera el inicio de la primera fase del boom de Internet, que en poco más de cinco años acabó con la explosión de la burbuja financiera que había sido generada por las expectativas de negocio de las primeras empresas. Durante los primeros años de Internet hubo una gran revolución tecnológica que hizo que Internet, y más exactamente la World Wide Web (o simplemente la Web), el sistema de hipertexto que nos permite navegar a través de la información que circula por la Red de una forma sencilla y transparente, revolucionara nuestras vidas y la de muchas industrias para siempre.

Poco menos de diez años después, en la primera conferencia de O'Reilly Media, Tim O'Reilly, su fundador, acuñaba el termino Web 2.0 para referirse a la nueva revolución que estaba sufriendo Internet. Sin embargo, esta vez la revolución no hace referencia sólo a la tecnología (de hecho, la Web 2.0 no introduce ninguna especificación nueva sobre lo que es la Web, cosa que ha hecho que el creador original de la Web, Tim Berners-Lee haya cuestionado lo apropiado del nombre), sino también a las personas.

La rápida evolución del Internet ha desencadenado nuevos modelos de producir y compartir información. La Web, con un ámbito de influencia global, tiene como su más valiosa contribución acercar a los usuarios por encima de fronteras geográficas, permitiendo la comunicación entre personas, más que entre computadoras. Pasando por la Web 1.0, donde las posibles tareas de los usuarios se limitaban a buscar y consultar la información de diferentes sitios web, generalmente estáticos y muchos de ellos elaborados por especialistas, la Web ha evolucionado de tal manera que son los mismos usuarios los que participan en la construcción de los contenidos.

El Web 2.0 se basa en la creación de una web enfocada al usuario y orientada a la interacción como las redes sociales. Es decir, los sitios web 2.0 actúan como puntos de encuentro o páginas webs que dependen directamente de los usuarios. Lo que se pretende es que sea el usuario el que cree contenidos web gracias a la utilización de los servicios de las páginas. Por lo tanto, los sitios web dejan de tener sentido sin usuarios que exploten los servicios que éste ofrece, ahora los usuarios pasan de tener una actividad pasiva a activa, donde pueden participar aportando contenidos o recursos a la aplicación web [4].

Figura 1. Aplicaciones Web 2.0

La Web 2.0 e Internet en general están rompiendo paradigmas de hábitos, negocios, políticas... en muchos ámbitos: prensa, industria musical, industria televisiva, industria cinematográfica, fotografía, política. En particular destaca la capacidad de la Web 2.0 para impulsar un nuevo paradigma creativo, para convertirse en el motor que necesitaba lo que algunos gurús como Charles Leadbeater han denominando *mass creativity* (creación en masa). Una suerte de creatividad social en la que destaca la participación del usuario como "prosumidor" (otro neologismo de la industria de la tecnología que se refiere a personas que están a medio camino entre un consumidor y un profesional, por ejemplo los fotógrafos aficionados que sin ser profesionales de la fotografía van más allá en su uso y habilidad con la cámara que un consumidor medio) y la formación de comunidades de innovación con un alto nivel de auto organización.

Es otro reflejo de la revolución social invisible en la que el usuario tiene a su disposición las herramientas para convertirse fácilmente en productor, y sobre todo tiene a su disposición un medio de tanto alcance como es Internet. Pero si algo hay que destacar de esta creatividad en masa propiciada por la Web 2.0 es el ejemplo de la co-creación.

Una co-creación que va desde lo más simple –como la concentración de pequeñas muestras de arte y creatividad del usuario aficionado que permiten sitios Web 2.0 como YouTube o Flickr–, pasando por la aportación del trabajo individual en pequeñas piezas para dar lugar a un todo inabarcable, no ya por una única persona sino por las mayores empresas del mundo. En este caso es la Wikipedia el ejemplo más emblemático, con sus mareantes números de entradas, aportaciones y capacidad de corrección y puesta al día. La Wikipedia es una enciclopedia multilingüe en formato on line creada, editada y corregida mediante la contribución de los propios usuarios (y no con un grupo de expertos como las enciclopedias tradicionales) sobre una plataforma tecnológica conocida como wiki que permite de forma fácil y dinámica esta edición y creación conjunta de una forma descentralizada. Todo ello gracias a la creación en masa de cientos de miles de pequeños contribuyentes y a la tarea de revisor de un millar de editores.

En definitiva, en un mundo de continuos cambios creativos, la Web 2.0 propicia que lo único constante sea la innovación.

El concepto Web 2.0 abarca una serie de aplicaciones que proporcionan servicios interactivos en red proporcionando al usuario el control de sus datos. Las redes sociales, con su trabajo colaborativo como estandarte; los blogs, con su valiosa retroalimentación en la comunicación; la sindicación de contenidos, con la agilización en la recepción de información útil; y los wikis, de los que hemos hablado anteriormente, son algunas de las aplicaciones Web 2.0 que tenemos a nuestra disposición.

Debido a que el proyecto que se ha realizado entra en el marco de las redes sociales, a continuación entraremos en detalle sobre el impacto que han tenido las mismas en la sociedad.

2.2.1. Redes sociales

Antes de la explosión de la web, el círculo de personas al que podías acceder para compartir tus intereses, tus visiones de la vida o sencillamente tu quehacer diario estaba limitado al espacio físico y temporal en el que te movías, y eso hacía que tu número de interacciones sociales fuera limitado.

Con la emergencia de Internet y sobre todo de la Web 2.0 mediante las llamadas redes sociales, la limitación espacial física/presencial para las interacciones sociales desaparece y la temporal se hace mucho más manejable por la posibilidad de la asincronía en la comunicación. Hoy en día, seis de los diez sitios más visitados en la Web son redes sociales.

Las redes sociales son espacios de Internet donde las personas publican y comparten todo tipo de información, personal y profesional, con terceras personas. De este modo, las redes sociales favorecen el trabajo colaborativo y la democratización de los contenidos, siendo el lugar propicio para que los usuarios desarrollen la propia Red.

Figura 2. Principales redes sociales

La tecnología sobre la que se sustentan las redes sociales permite a sus usuarios compartir todo tipo de datos e información en múltiples formatos (audio, texto y vídeo). De esta manera los usuarios pueden crear nuevos contenidos, mezclar los contenidos existentes, agregar comentarios, etiquetar diferentes contenidos, y finalmente compartir los contenidos con usuarios de otras partes del mundo, entrando a un ciclo permanente de retroalimentación.

Han surgido hasta hoy un gran número de redes sociales en Internet, entre las cuales podemos mencionar Facebook, LinkedIn, MySpace, Twitter, Slashdot, Reddit, Digg, Delicious, StumbleUpon, FriendFeed, Last.fm, Friendster, LiveJournal, Hi5, Tagged, Ning, Xanga, Classmates y Bebo.

No obstante es importante señalar que las redes sociales no solo aportan beneficios. Las redes sociales pueden representar un serio problema en cuanto a la privacidad y/o

confidencialidad. Que los datos personales de los usuarios se encuentren a disposición de cualquier persona no deja de representar un riesgo.

En cualquier caso, es muy estimulante y cuando menos interesante contemplar el ritmo de creación e innovación constante en la tecnología y en las relaciones sociales que las redes sociales han propiciado [5].

2.3. Tecnologías basadas en Web 2.0

El Web 2.0 no pretende exigir la utilización de unas determinadas tecnologías para que todas nuestras aplicaciones web entren en este esquema, sino más bien pretende crear una tendencia. Sin embargo, existen varias tecnologías que están utilizándose actualmente en busca de seguir evolucionado el Web. Algunas tecnologías y directrices que dan vida a un proyecto Web 2.0 son:

- Transformar software de escritorio hacia la plataforma web.
- Respeto a los estándares del W3C.
- Separación del contenido del diseño con uso de hojas de estilo CSS.
- Sindicación y agregación de contenidos (RSS / ATOM).
- Interacción asíncrona con AJAX
- Utilización de redes sociales al manejar usuarios y comunidades.
- Dar control total a los usuarios en el manejo de su información.
- Proveer APIs o XML para que las aplicaciones puedan ser manipuladas por otros.
- Facilitar el posicionamiento con URL´s sencillos.

Dentro del proyecto utilizaremos tecnologías que siguen la tendencia Web 2.0 como lo son AJAX, CSS y Google Maps API.

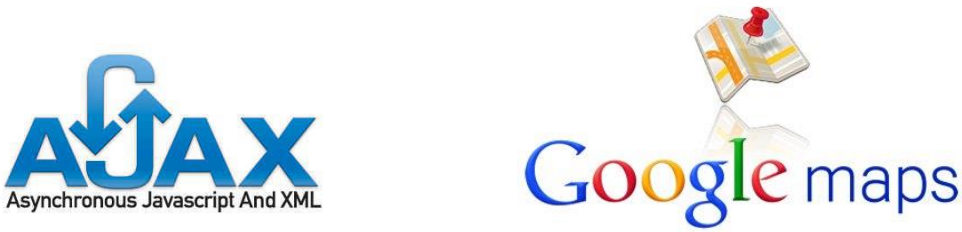

Figura 3. Tecnologías Web 2.0

2.3.1. AJAX

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el lado del cliente, o mejor dicho en el navegador del usuario, y mantiene una comunicación asíncrona con el servidor en segundo plano. Esto tiene la ventaja de realizar cambios sobre la misma página sin necesidad de recargar todo el contenido de la misma, como ocurre cuando pinchamos en un enlace o cuando pulsamos el botón *submit* de un formulario. Esto puede traducirse como un aumento de la interactividad, velocidad y usabilidad de la aplicación web.

Esta tecnología está basada en el objeto XMLHttpRequest, el cual es un objeto suministrado por los navegadores webs, accesible mediante JavaScript, usado para transferir datos asíncronos entre el navegador y el servidor web [6].

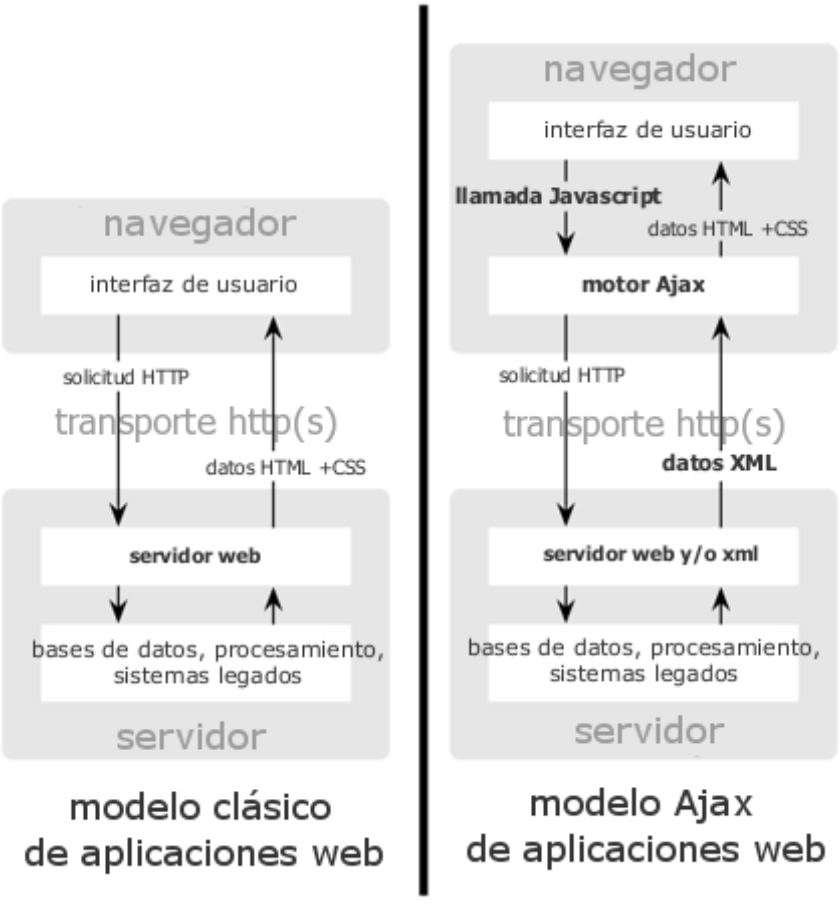

Figura 4. Comparativas entre modelos de aplicaciones web

El formato JSON es el más adecuado para la respuesta del servidor cuando la acción AJAX debe devolver una estructura de datos a la página que realizó la llamada de forma que se pueda procesar con JavaScript.

JSON (JavaScript Object Notation) es un formato sencillo para intercambiar datos. Consiste básicamente en un array asociativo de JavaScript que se utilizar para incluir información del objeto. JSON ofrece 2 grandes ventajas para las interacciones AJAX: es muy fácil de leer en JavaScript y puede reducir el tamaño en bytes de la respuesta del servidor [7].

2.3.2. CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes [8].

Figura 5. Código CSS

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

2.3.3. Google Maps API

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Google y que ofrece imágenes de mapas desplazables [9].

Google Maps fue una de las primeras aplicaciones basadas en AJAX de uso masivo por parte de los usuarios. En realidad es sólo HTML, CSS y JavaScript trabajando junto. Los mapas son imágenes que se cargan en el fondo a través de peticiones ejecutadas por la tecnología de AJAX, y se insertan en un en la página HTML. Mientras se navega en el mapa, el API envía información acerca de las nuevas coordenadas y los niveles de "zoom" del mapa a través de AJAX y esto retorna las imágenes.

Figura 6. Imágen de Google Maps

Google ofrece de forma gratuita una API con la que poder desarrollar aplicaciones a medida basadas en los mapas de Google, integrar los mapas en otras aplicaciones e incluso hacer "mash-up"o mezclas de Google Maps y otras aplicaciones web que también disponen de una API pública.

El API consiste de archivos JavaScript que contienen las clases, métodos y propiedades que se usan para el comportamiento de los mapas.

2.4. J2EE – Tecnología de desarrollo del proyecto

La decisión de utilizar la tecnología J2EE como la herramienta de trabajo para la realización del proyecto ha sido porque se trata de una tecnología *Open Source* de la cual existen multitud de herramientas gratuitas. Además incluye una gran cantidad de documentación y API´s superiores a las que puedan existir para otras plataformas como .NET y esto es un factor muy importante, ya que facilita y agiliza el diseño e implementación del proyecto. También es interesante fomentar el uso de tecnologías libres a nivel empresarial y poder demostrar que se pueden realizar proyectos sin necesidad de tener que invertir en herramientas que en muchos casos pueden resultar extremadamente costosas.

Así que los motivos por los cuales se ha elegido J2EE como tecnología de desarrollo han sido:

- Conocimiento previo del lenguaje Java.
- Cubre grandes necesidades tecnológicas como páginas dinámicas JSP y lógica de negocio mediante JAVA.
- Interoperable con otras tecnologías como XML, JavaScript, HTML…
- Soporte: API´s, manuales, ejemplos…
- *Open Source* y herramientas de desarrollo como lo es Eclipse.
- Muchas utilidades ya creadas y fáciles de integrar.
- Existencia de una gran variedad de frameworks MVC para el desarrollo de aplicaciones, entre ellos Spring, que es el objetivo del proyecto.

2.5. Frameworks J2EE

Los frameworks están de moda en la actualidad en el mundo de la Ingeniería del Software.

Un framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado.Puede incluir soporte de programas, bibliotecas y un lenguaje de *scripting* para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Por lo tanto, es una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.Para ello, es indispensable conocer su metodología de trabajo.

En el momento de crear una nueva aplicación de software, los programadores tienen diferentes alternativas para elegir el framework que utilizarán para desarrollar su trabajo. En el mundo de la tecnología J2EE, los frameworks más conocidos son: Struts,

Spring, Jsf, entre otros, y que han nacido como idea de un pequeño grupo de expertos y que han logrado llamar la atención del mundo de la informática.

Figura 7. Ejemplo de Frameworks J2EE

En el anexo B, se puede ver una comparativa de los principales frameworks del mercado.

Es importante decir que los frameworks, a pesar de hacer más fácil la vida del programador, tienen una curva larga de aprendizaje. Pero una vez superada esta barrera, la herramienta permitirá que los programadores puedan optimizar la lógica de programación sin preocuparse de otros detalles más simples del sistema.

Las principales características que un framework ofrece son:

- Automatizar tareas comunes como validaciones y conversiones de datos.
- Acelerar e incrementar el proceso de desarrollo.
- Reutilizar el código ya existente.
- Promover buenas prácticas de desarrollo como el uso de patrones

Debido a que Spring es el framework que se ha elegido para desarrollar el proyecto, es conveniente ver con más detalle sus principales características.

2.6. Spring 3

En este apartado se dan a conocer todos los aspectos técnicos de Spring, los conceptos básicos, sus características esenciales, las partes de su arquitectura,así como los componentes que lo conforman.

2.6.1. Introducción e historia

Spring es un framework de aplicación desarrollado por la compañía Interface 21,para aplicaciones escritas en el lenguaje de programación Java.Fue creado gracias a la colaboración de grandes programadores,entre ellos se encuentran como principales partícipes y líderes de este proyecto Rod Johnson y Jürgen Höller.Estos dos desarrolladores,además de otros colaboradores que juntando toda su experiencia en el desarrollo de aplicaciones J2EE (incluyendo EJB ,Servlets y JSP), lograron combinar dichas herramientas y otras más en un solo paquete,para brindar una estructura más sólida y un mayor soporte para este tipo de aplicaciones.

Figura 8. Logo de Spring

Spring nos proporciona varios módulos los cuales abarcan la mayor parte de las tareas que debemos hacer en cualquiera de las capas de nuestras aplicaciones, desde plantillas para trabajar con JDBC o invocación de Web Services y JMS, pasando por sus propias soluciones ORM o MVC (web), hasta integración con otros frameworks, como Struts 2, Hibernate, JSF, etc. Todo esto de una forma elegante y haciendo uso de muchos buenos principios de programación. Además Spring maneja la infraestructura de la aplicación, por lo que nosotros sólo deberemos preocuparnos de la lógica de la misma (y de la configuración de Spring).

Spring es, como lo definen sus autores, un framework ligero para construir aplicaciones empresariales.La separación en módulos nos permite usar sólo las partes que necesitamos, sin tener la carga de los que no usemos.

Spring está diseñado para no ser intrusivo, esto significa que no es necesario que nuestra aplicación extienda o implemente alguna clase o interface de Spring (si no lo queremos), por lo que nuestro código de lógica quedará libre y completamente reutilizable para un proyecto sin Spring, o por si debemos quitarlo de una aplicación que

ya lo esté usando. Gracias a esto es posible usar un POJO o un objeto Java para hacer cosas que antes solo podían hacerse con EJBs. Sin embargo la utilidad de Spring no es sólo para el desarrollo de aplicaciones web. Cualquier aplicación Java puede beneficiarse del uso de Spring.

Además, si usamos Spring de la forma correcta nuestra aplicación quedará dividida en capas bien delimitadas, y con buenas prácticas de programación.

Este framework está adquiriendo gran auge y una gran popularidad.Una de las características que ayuda a este éxito, es que es una aplicación *open source*, lo cual implica que no tienen ningún coste, ni se necesita una licencia para utilizarlo, por lo tanto da libertad a muchas empresas y desarrolladores a incursionar en la utilización de esta aplicación. Además de que está disponible todo el código fuente de este framework en el paquete de instalación.

En general, estas son algunas de las características de Spring:

- Simplificación de la programación orientada a aspectos.
- Simplificación del acceso a datos.
- Simplificación e integración con JEE
- Soporte para planificación de trabajos.
- Soporte para envio de mail.
- Interacción con lenguajes dinámicos (como BeanShell, JRuby, y Groovy).
- Soporte para acceso a componentes remotos.
- Manejo de transacciones.
- Su propio framework MVC.
- Su propio Web Flow.
- Manejo simplificado de excepciones.

2.6.2. Novedades de Spring 3

La versión 3 de Spring es una versión revisada y mejorada de la versión estable anterior (2.5), que incluye nuevas características, entre las que se incluyen:

- Soporte para Java 5: proporciona configuración basada en anotaciones, además la parte web es compatible con las versiones 1.4 y 5 de Java EE. Debido a esta nueva característica, ahora es necesario tener el JRE versión 5 o superior.
- Lenguaje de Expresiones (SpEL): en esta nueva versión se incluye un lenguaje de expresiones que puede ser usado cuando se definen beans, tanto en XML como con anotaciones y también da soporte a través de todos los módulos de Spring.
- Soporte para Servicios Web REST: ahora Spring soporta servicios web de tipo REST.

- Soporte para Java EE6: ofrece soporte de características como JPA 2.0, JSF 2.0 y JRS 303 (validaciones de Beans).
- Soporte para bases de datos embebidas: un soporte conveniente para bases de datos embebidas como HSQL, H2 y Derby.
- Soporte para formateo de datos mediante anotaciones: ahora los campos de fecha, divisas, etc., serán formateados automáticamente y convertidos usando anotaciones.
- Nueva organización de los módulos: los módulos han sido revisados y separados en diferentes paquetes, más organizados, de acuerdo a su funcionalidad. [10]

2.6.3. Arquitectura de Spring

Spring es un framework modular que cuenta con una arquitectura organizada en 20 módulos. Estos módulos están agrupados en los siguientes grupos:

- Contenedor Central (Core Container): nos permite configurar nuestra aplicación, incluyendo la caracterísitica de inyección de dependencias para desacoplar el código de nuestra aplicación.
- Acceso a Datos / Integración: permite la integración de la persistencia en nuestras aplicaciones.
- WEB: permite la integración del desarrollo de aplicaciones orientadas al web.
- AOP (Programación Orientada a Aspectos): permite la integración del desarrollo de aplicaciones orientadas a aspectos.
- Instrumentación: proporciona una implantación de cargador de clases para ciertos servidores de aplicaciones.
- Pruebas: permite la integración de componentes de testeo, tal como Junit en el desarrollo de aplicaciones.

Estos grupos se muestran en la siguiente imagen:

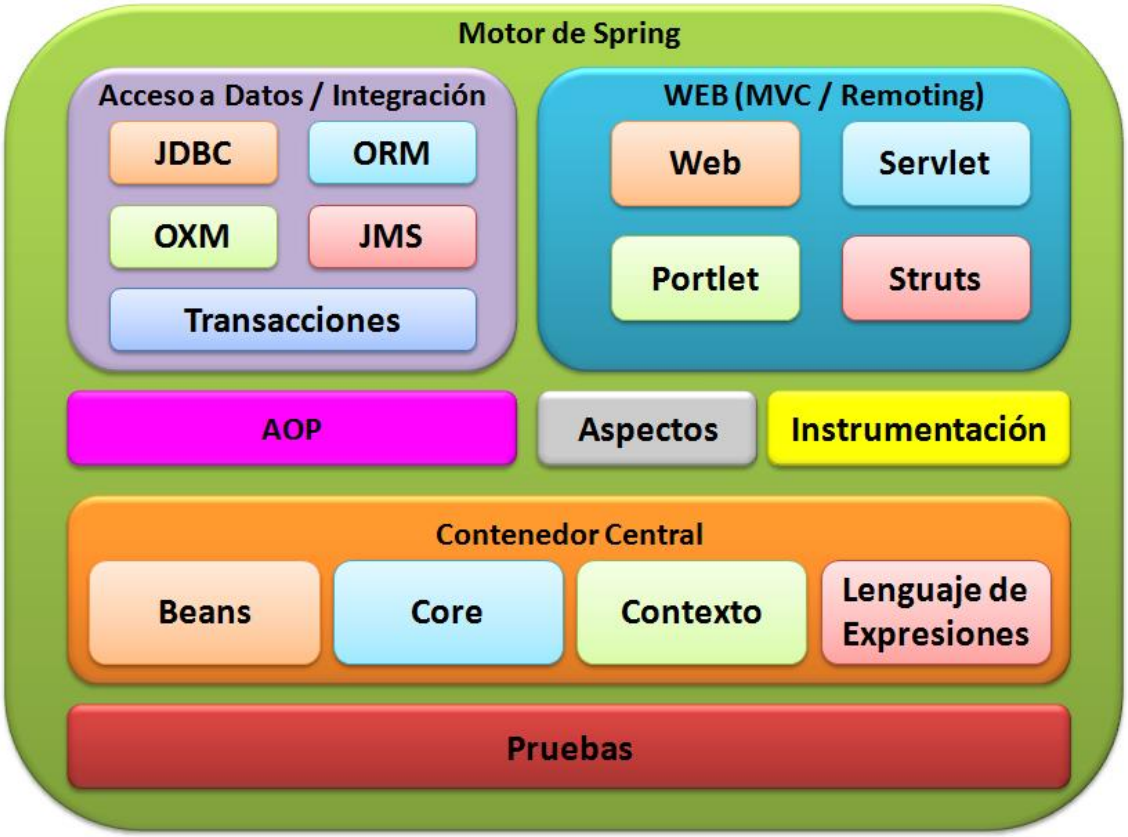

Figura 9. Arquitectura de Spring

A continuación, nos centraremos en los módulos más importantes de Spring.

2.6.3.1. Contenedor Central

El contenedor de Spring es uno de los puntos centrales de Spring, provee la funcionalidad esencial del framework.

En Spring, los objetos que forman la columna vertebral de las aplicaciones, y que son administradas por el contenedor de Spring, son llamados "beans". Un bean es un objeto que es instanciado, ensamblado (cuando sus dependencias son inyectadas), y en general, administrado por el contenedor. Los beans, y las dependencias entre ellos, las declaramos en los metadatos del contenedor de Spring (lo cual puede ser mediante anotaciones o archivos de mapeo).

En las aplicaciones basadas en Spring, los objetos de la aplicación viven dentro del contenedor de beans. El contenedor se encarga de crear los objetos y "cablearlos", configurarlos, y administrar su ciclo de vida completo.

Hay dos tipos de contenedores de beans:

BeanFactory

Representa las fábricas de beans y son el tipo de contenedor más simple. Proporciona la funcionalidad básica. Como un BeanFactory conoce a los objetos que componen nuestra aplicación (los declarados en el archivo de configuración), es capaz de crear las asociaciones entre estos objetos en el momento de ser instanciados. Además, como el BeanFactory tiene el control del ciclo de vida del bean, puede hacer llamadas a métodos personalizados de inicialización y destrucción (si estos métodos están definidos).

En el momento en el que creamos nuestro BeanFactory, éste lee las definiciones de los beans del archivo indicado, sin embargo los beans no son creados en ese momento. Los beans se crean cuando que son necesitados.

Application Context

Un BeanFactory está bien para aplicaciones simples, pero no toma ventaja de todo el poder que nos proporciona el framework de Spring. Para ello tenemos un contenedor más avanzado que representa el contexto de la aplicación.

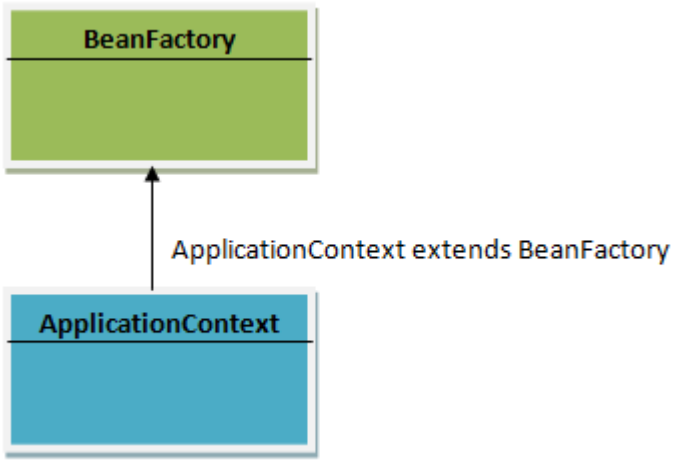

Figura 10. Contenedores de beans

Usar ApplicationContext es similar a usar BeanFactory. Ambos cargan definiciones de beans, ensamblan los beans, y los envían a quienes se los piden. El contenedor obtiene la información de qué objetos debe crear leyendo los metadatos de la configuración, los cuales, pueden ser colocados en archivos XML, anotaciones, o dentro del código Java.

ApplicationContext ofrece más cosas:

- Proporciona un medio para resolver mensajes de texto, incluyendo soporte para internacionalización (I18N) de estos mensajes.
- Proporciona una manera genérica de cargar archivos de recursos, como imágenes.
- Puede publicar eventos a beans que están registrados como listeners.
- Puede cargar múltiples contextos

Debido a que ApplicationContext proporciona más funcionalidades, es preferible el uso de éste sobre BeanFactory en casi todo tipo de aplicaciones, la única excepción es cuando estamos ejecutando una aplicación donde los recursos son escasos, como en un dispositivo móvil.

Por defecto todos los beans manejados por Spring son singletons, esto es, sólo existe una instancia de ellos por aplicación.ApplicationContext carga todos los singletons cuando se inicia el contexto, y de esta forma se asegura de que estarán listos en el momento en el que sean solicitados.

Ahora vamos a analizar de las 2 características más importantes de Spring:

Inversión de Control

El núcleo de Spring está basado en un principio o patrón de diseño llamado Inversión de Control (IoC por sus siglas en inglés). Las aplicaciones que usan el principio de IoC se basan en su configuración (que en este caso puede ser en archivos XML o con anotaciones como en Hibernate) para describir las dependencias entre sus componentes, esto es, los otros objetos con los que interactúa.

La aplicación no controla su estructura; permite que sea el framework de IoC (en este caso Spring) quien lo haga.En este caso en vez de ser el mismo objeto quien se encargue de instanciar, o localizar las dependencias con las que trabaja (usando directamente su constructor o un localizador de servicios), es el contenedor el que inyecta estas dependencias cuando crea al bean.

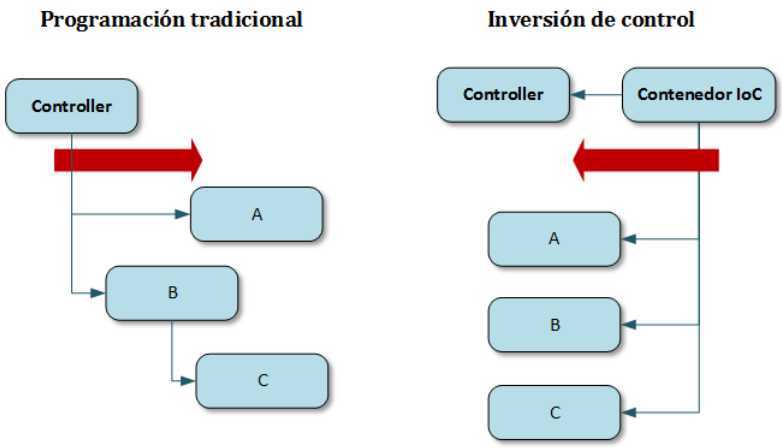

Figura 11. Ejemplo de inyección de dependencias

Este proceso, como podemos observar, es el inverso al que normalmente se hace, y de ahí el nombre de Inversión de Control (es el framework el que hace el trabajo, no el programador.De esta manera el programador pasa a limitar su responsabilidad a la configuración de los beans que luego Spring se encargará de crear.

Inyección de dependencias

En Spring, los objetos no son responsables de encontrar o crear los otros objetos que necesitan para hacer su trabajo. En vez de eso, el contenedor les da las referencias de los objetos con los que colaborarán.

En el contexto de DI, Spring actúa como un contenedor que proporciona las instancias de las clases que nuestra aplicación necesita, pero en una forma no intrusiva y automática. [11]

Todo lo que debemos hacer es crear un archivo de configuración que describa las dependencias, y Spring se hará cargo del resto.O mediante la anotación @autowired le indicamos a Spring que se tiene que encargar de buscar un bean que cumpla los requisitos para ser inyectado.En caso de que hubiese más de un bean que cumpliese esos requisitos tendríamos que decirle a Spring cuál es el correcto.

Sin inyección de dependencias, cada clase llama al objeto que necesita en tiempo de ejecución. Mientras que con inyección de dependencias, cada objeto es cargado en cada clase que lo necesita en tiempo de inicialización.

La implementación de DI de Spring se enfoca en el acoplamiento débil: los componentes de nuestra aplicación deben asumir lo menos posible acerca de otros componentes. La forma más fácil de lograr este bajo acoplamiento en Java es mediante el uso de Interfaces. Cada componente de la aplicación sólo es consciente de la interface de otros componentes, por lo que podemos cambiar la implementación sin afectar a los otros componentes.

El uso de interfaces y DI son mutuamente benéficos, ya que hace más flexible y robusta nuestra aplicación y es mucho más fácil realizar pruebas unitarias.

El uso de DI tiene como beneficios, además de los mencionados antes, los siguientes:

- Reduce el código pegamento: reduce dramáticamente la cantidad de código que debemos escribir para unir los distintos componentes. Aunque algunas veces este código puede ser tan simple como usar el operador "new" para instanciar un nuevo objeto, otras puede ser más complejo, como realizar una búsqueda de dicha dependencia en un repositorio a través de JNDI, como en el caso de los recursos remotos. En este caso, el uso de DI puede reducir de forma dramática la cantidad de código pegamento proporcionando búsquedas automáticas.

- Externaliza dependencias: como es posible colocar la configuración de dependencias en archivos XML podemos realizar una reconfiguración fácilmente, sin necesidad de recompilar nuestro código. Gracias a esto es posible realizar el cambio de la implementación de una dependencia a otra .

- Las dependencias se manejan en un solo lugar: toda la información de dependencias es responsabilidad de un sólo componente, el contenedor de IoC de Spring, haciendo este manejo de dependencias más simple y menos propenso a errores.

- Hace que las pruebas sean más fáciles: nuestras clases serán diseñadas para que sea fácil el reemplazo de dependencias. Podemos proporcionar *dummies*, que regresen datos de prueba, de servicios o cualquier dependencia que necesite el componente que estamos probando.

Como podemos ver, el uso de DI nos proporciona muchos beneficios, pero no sin sus correspondientes desventajas. En particular, es difícil ver qué implementación particular de una dependencia está siendo usada para qué objeto, especialmente para alguien que no está familiarizado con esta forma de trabajo.

2.6.3.2. Acceso a datos/Integración

Spring aborda los problemas comunes que enfrentan los desarrolladores al trabajar con bases de datos en las aplicaciones. Se proporciona soporte para las herramientas de acceso de datos más populares en Java: JDBC, iBatis/MyBatis, Hibernate, JDO, JPA, Oracle TopLink, Apache OJB y Apache Cayenne, entre otros.[12]

Algunas de las ventajas que brinda Spring al combinarse con alguna herramienta ORM son:

- Manejo de sesión: Spring hace de una forma más eficiente, sencilla y segura la forma en que se manejan las sesiones de cualquier herramienta ORM que se quiera utilizar.
- Manejo de recursos: se puede manejar la localización y configuración de los SessionFactories de Hibernate o las fuentes de datos de JDBC haciendo que estos valores sean más fáciles de modificar.
- Manejo de transacciones integrado: se puede utilizar una plantilla de Spring para las diferentes transacciones ORM.
- Envolver excepciones: con esta opción se pueden envolver todas las excepciones para evitar el tratamiento de excepciones en cada segmento de código necesarios.
- Evita limitarse a un sólo producto: si se desea migrar o actualizar a otra versión de un ORM distinto o del mismo,Spring trata de no crear una dependencia entre la herramienta ORM, el mismo Spring y el código de la aplicación , para que cuando sea necesario migrar a un nuevo ORM no sea necesario realizar tantos cambios.
- Facilidad de prueba: Spring trata de crear pequeños pedazos que se puedan aislar y probar por separado, ya sean sesiones o una fuente de datos.

El patrón DAO (Data Access Object) es uno de los patrones más importantes y usados en aplicaciones J2EE, y la arquitectura de acceso a los datos de Spring provee un buen soporte para este patrón, proporcionando plantillas que nos ahorrarán muchas líneas de código.Las plantillas gestionan las partes fijas del acceso a datos, controlan las excepciones, asignan los recursos y manejan las transacciones.

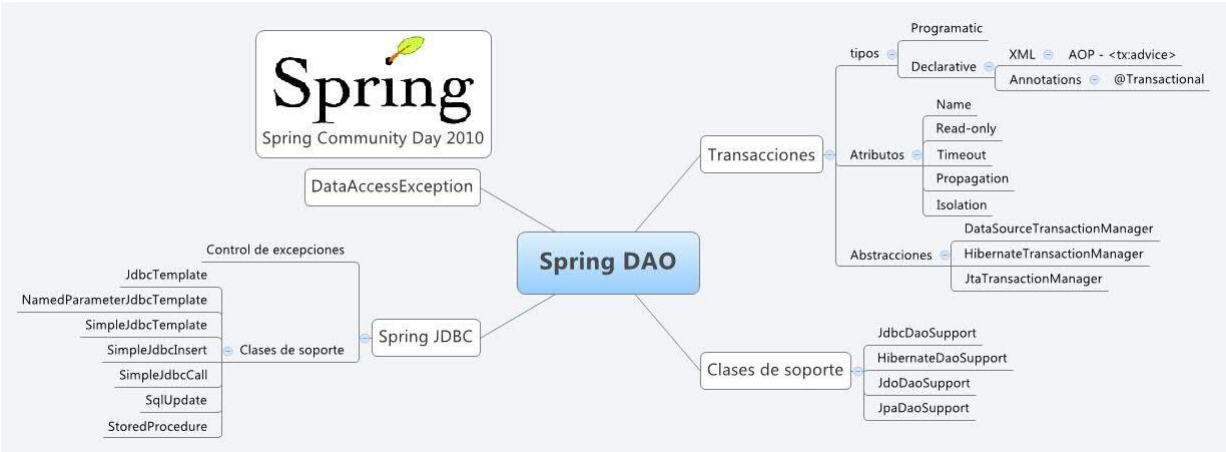

Figura 12. Soporte de Spring para el patrón DAO

Spring tiene varias plantillas dependiendo de la tecnología de persistencia que vayamos a usar. Usar una plantilla simplifica mucho el código de acceso a datos y se configura como un bean del contexto de Spring.

Spring facilita la construcción de DAO en Hibernate, mediante la clase HibernateDaoSupport. Esta clase nos permite inyectar en nuestros DAOs nuestro SessionFactory de Hibernate o la plantilla HibernateTemplate. La plantilla HibernateTemplate simplifica trabajar con el objeto Session de Hibernate, siendo responsable de abrir y cerrar sesiones y gestionar excepciones principalmente.

Mediante el uso de esta clase de soporte Spring facilita al máximo la codificación de nuevas clases, reduciendo el número de líneas de código de éstas y reduciéndolas a la mínima expresión.

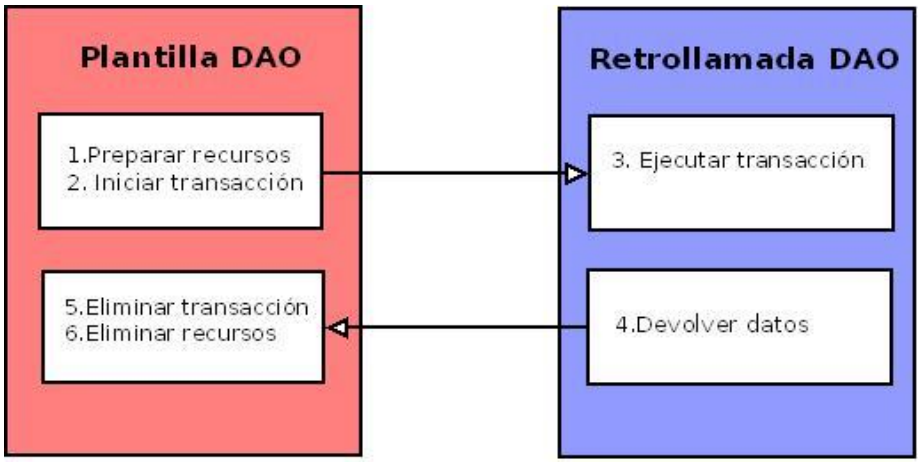

Figura 13. Tareas de la plantilla DAO

2.6.3.3. Spring AOP

Aspect-Oriented Programming (AOP) es un paradigma de programación que nos permite separar "aspectos" de nuestro software en diferentes módulos según su responsabilidad. De esta manera puede verse como otra herramienta para poder diferenciar en nuestro código las diferentes necesidades, encapsulando los diferentes conceptos como pueden ser los requerimientos funcionales, seguridad, *profiling* de tiempos, transaccionabilidad contra una base de datos, *logging*, entre otros.[13]

Los términos básicos utilizados en el mundo AOP son:

- Aspect (Aspecto) es una funcionalidad transversal (*cross-cutting*) que se va a implementar de forma modular y separada del resto del sistema. El ejemplo más común y simple de un aspecto es el *logging* (registro de sucesos) dentro del

sistema, ya que necesariamente afecta a todas las partes del sistema que generan un suceso.

- Join point (Punto de Cruce o de Unión) es un punto de ejecución dentro del sistema donde un aspecto puede ser conectado, como una llamada a un método, el lanzamiento de una excepción o la modificación de un campo. El código del aspecto será insertado en el flujo de ejecución de la aplicación para añadir su funcionalidad.
- Advice (Consejo) es la implementación del aspecto, es decir, contiene el código que implementa la nueva funcionalidad. Se insertan en la aplicación en los "Puntos de Cruce".
- Pointcut (Puntos de Corte) define los "Consejos" que se aplicarán a cada "Punto de Cruce". Se especifica mediante expresiones regulares o mediante patrones de nombres (de clases, métodos o campos), e incluso dinámicamente en tiempo de ejecución según el valor de ciertos parámetros.
- Introduction (Introducción) permite añadir métodos o atributos a clases ya existentes. Un ejemplo en el que resultaría útil es la creación de un "Consejo" de auditoría que mantenga la fecha de la última modificación de un objeto, mediante una variable y un método setUltimaModificacion(fecha), que podrían ser introducidos en todas las clases (o sólo en algunas) para proporcionarlas esta nueva funcionalidad.
- Target (Destinatario) es la clase aconsejada, la clase que es objeto de un consejo. Sin AOP, esta clase debería contener su lógica, además de la lógica del aspecto.
- Proxy (Resultante) es el objeto creado después de aplicar el "Consejo" al "Objeto Destinatario". El resto de la aplicación únicamente tendrá que soportar al "Objeto Destinatario" (pre-AOP) y no al "Objeto Resultante" (post-AOP).
- Weaving (Tejido) es el proceso de aplicar "Aspectos" a los "Objetos Destinatarios" para crear los nuevos "Objetos Resultantes" en los especificados "Puntos de Cruce". Este proceso puede ocurrir a lo largo del ciclo de vida del "Objeto Destinatario":
    - "Aspectos" en tiempo de compilación.
    - "Aspectos" en tiempo de carga: los "Aspectos" se implementan cuando el "Objeto Destinatario" es cargado.
    - "Aspectos" en tiempo de ejecución.

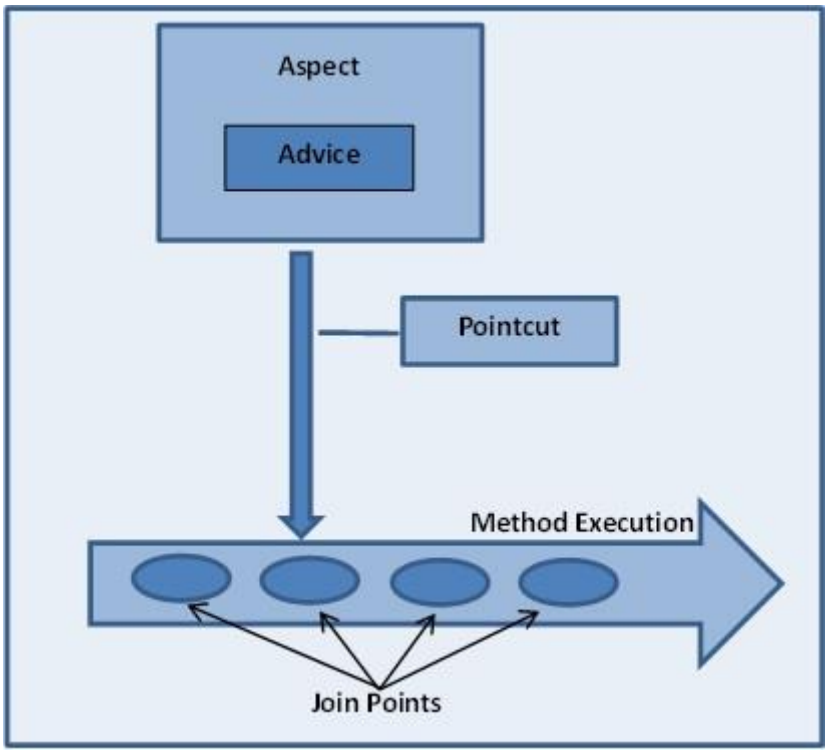

Figura 14. Conceptos AOP

2.6.3.4. Spring Web MVC

Spring MVC es uno de los módulos del framework Spring, y provee un exhaustivo soporte para el patrón MVC.

El Web MVC de Spring presenta algunas similitudes con otros frameworks para web que existen en el mercado, pero tiene características que lo vuelven único:

- Provee una limpia separación entre el modelo de dominio y los formularios web y lo integra con todas las demás características de Spring Framework.
- Spring puede ser fácilemnte integrado con otros frameworks MVC.
- Spring brinda un MVC para web bastante flexible y altamente configurable, pero esta flexibilidad no le quita sencillez, ya que se pueden desarrollar aplicaciones sencillas sin tener que configurar muchas opciones.
- Spring brinda soporte para JSP, Struts, Velocity, entre otros.

Para intentar comprender cada parte de la arquitectura del Web MVC de Spring analizaremos el ciclo de vida de una petición o request.[14]

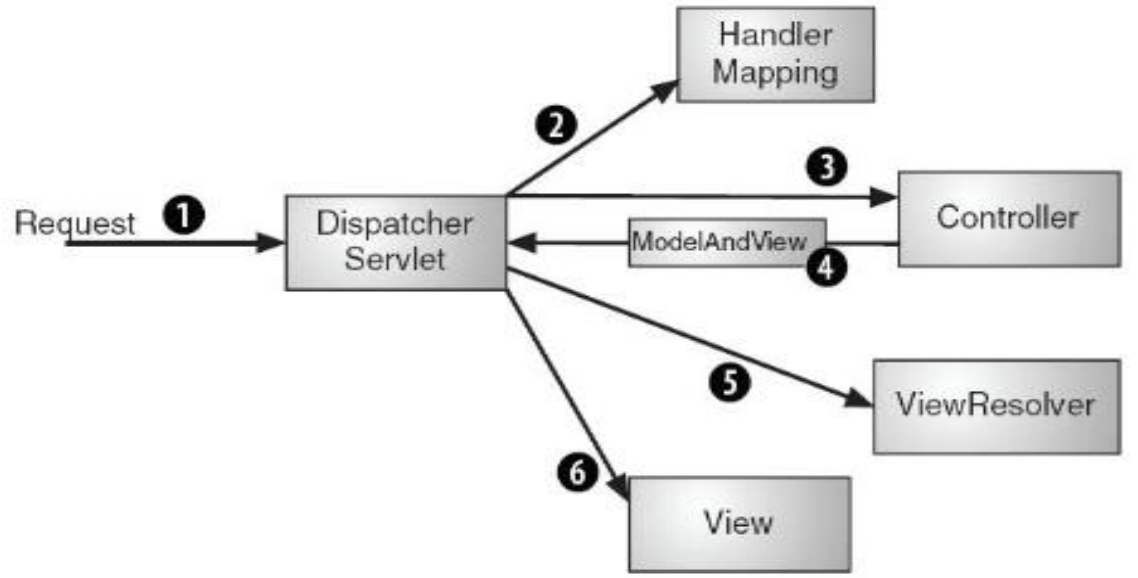

Figura 15. Ciclo de vida de una petición

1. El navegador manda una petición y lo recibe un DispatcherServlet.

2. El DispatcherServlet debe escoger qué controlador manejará la petición. Para ello, el HandlerMapping mapea los diferentes patrones de URL hacia los controladores, y le comunica al DispatcherServlet el controlador elegido.

3. El controlador elegido toma la petición y ejecuta la tarea.

4. El controlador retorna un objeto ModelAndView al DispatcherServlet.

5. Si el ModelAndView contiene un nombre lógico de una vista se utiliza un ViewResolver para buscar ese objeto vista que mostrará la respuesta de la petición.

6. Finalmente el DispatcherServlet llama a la vista adecuada.

Vamos a analizar con más profundidad varios de los componentes que participan en el ciclo de vida de la petición.[15]

DispatcherServlet

En Spring MVC el DispatcherServlet es un Servlet que recibe las peticiones HTTP y las envía al controlador apropiado. En una aplicación Spring MVC puede haber varios DispatcherServlet para cumplir con varios propósitos (por ejemplo manejar las solicitudes de las interfaces de usuarios, solicitudes de webServices, etc.) y cada DispatcherServlet tiene su propia configuración (WebApplicationContext), el cual define las características del Servlet tales como los controladores que el Servlet soporta, manejador de mapeo, etc.

También el WebApplicationContext puede incluir configuración de la aplicación como indicar la capa de persistencia, seguridad, servicios, etc.

El DispatcherServlet, como cualquier otro Servlet en una aplicación Java EE debe ser cargado en el tiempo de arranque de WEB-INF/web.xml. El DispatcherServlet también es responsable de cargar un SpringApplicationContext el cual es usado para realizar el enlazado y la inyección de dependencias.[16]

HandlerMapping

Existen diversas maneras con las que el DispatcherServlet puede saber qué controlador es el encargado de procesar una petición, y a qué bean del Application Context se lo puede asignar. Esta tarea la lleva cabo el Handler Mapping o el manejador de mapeo. Existen 2 tipos principales que son los que más se usan:

- BeanNameUrlHandlerMapping: mapea el URL hacia el controlador en base al nombre del bean del controlador.
- SimpleUrlHandlerMapping: mapea el URL hacia el controlador basado en una colección de propiedades declaradas en el applicationContext.

ViewResolver

Un View Resolver es el encargado de resolver el nombre lógico que retorna un controlador en un objeto ModelAndView, a un nombre de archivo físico que el navegador podrá desplegarle al usuario junto con los resultados procesados.

Spring MVC cuenta con cuatro View Resolvers diferentes, pero el más utilizado es InternalResourceViewResolver, el cual resuelve los nombres lógicos en un archivo tipo View que es convertido utilizando una plantilla de archivos como JSP, JSTL o Velocity.

Controladores

Los controladores retornan por lo general un objeto tipo ModelAndView, que es un objeto que contiene el modelo y la vista, ya que está compuesto de 2 o más atributos.

El objeto ModelAndView contiene el nombre lógico de la vista,el cual el framework se encarga (a través del View Resolver) de convertir ese nombre lógico al nombre físico, que es el que buscará el servidor web.

Además el modelo puede estar formado por un objeto o por un Map de objetos,los cuales se identificarán en la vista con el mismo nombre que se haya mandado dentro del modelo que se le dio al objeto ModelAndView.

También es bastante común que los controladores regresen un String en vez de un ModelAndView.El String contendría el nombre lógico de la vista antes citado.En este caso no se pasaría ningún modelo a la vista.

2.7. Hibernate

Para la mayoría de las aplicaciones, almacenar y recuperar información implica alguna forma de interacción con una fuente de datos. Esta fuente de datos normalmente es una base de datos relacional la cual presenta un problema para la gran mayoría de programadores puesto que el modelo relacional dista mucho del orientado a objetos, y no existe una estandarización para este manejo de datos.

Figura 16. Logo de Hibernate

En la actualidad el paradigma utilizado por la mayoría de analistas programadores a la hora de construir software es el orientado a objetos; independientemente de los avances en esta área (nuevos paradigmas), la base fundamental es este estilo de programación. Partiendo de esta afirmación toda aplicación estaría conformada por una gran cantidad de objetos interactuando y compartiendo información, y a la hora de que estos objetos tengan que persistir, inician una serie de problemas en la selección del modelo para persistir.[17]

Hibernate es un framework que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos. La idea de Hibernate la tuvo Gavin King (ingeniero de JBoss), el cual cansado de la ineficiencia y complejidad de los sistemas de persistencia de la época, ideó un sistema base que fue apoyado por un inmenso grupo de programadores alrededor del mundo. Esta idea de King se transformó en un proyecto robusto, con licencia libre que actualmente es el más utilizado, en cuanto a framework dedicados a persistencia se refiere.

2.7.1. Características

Entre las características más importantes de Hibernate se destacan:

- Software libre, distribuido bajo los términos de la licencia GNU LGPL.
- Soluciona el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional).
- Es flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente.
- Tiene la funcionalidad de crear la base de datos a partir de la información disponible.
- Ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").
- Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma y que veremos en detalle más adelante.

Para lograr solucionar la diferencia entre modelos de datos, Hibernate permite al desarrollador detallar cómo es su modelo de datos (utilizando metadatos que describen el mapeo entre objetos y la base de datos en sí), qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL.

Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate funciona asociando a cada tabla de la base de datos un objeto POJO (Plain Old/Ordinary Java Object). Un POJO es similar a un Java Beans, con propiedades accesibles mediante métodos setter y getter. Para poder asociar el POJO a su tabla correspondiente de la base de datos, se utilizan los ficheros hbm.xml. En estos ficheros se declaran las propiedades del POJO y sus correspondientes nombres de columna en la base de datos, asociación de tipos de datos, referencias, relaciones, etc.

Además,tenemos el archivo de propiedades “Hibernate.properties”,en el que podemos indicar qué gestor de bases de datos usaremos y a qué base de datos nos conectaremos y cómo lo haremos.

Con todo ello tenemos la ventaja de que nos es totalmente transparente el uso de la base de datos, pudiendo cambiar ésta sin necesidad de cambiar ninguna línea de código de la aplicación (sólo cambiando los ficheros de configuración).

Hibernate nos ofrece un lenguaje con el que poder realizar consultas a la base de datos. Este lenguaje es similar a SQL, y se denomina HQL. El HQL permite utilizar un lenguaje intermedio dependiendo de la base de datos que se utilice, y el dialecto será traducido al SQL dependiente de la base de datos de manera transparente y automática, y simplificando el código.

Hibernate soporta, entre otros, los siguientes dialectos: DB2, PostgreSQL, MySQL, Oracle, Oracle 9/10g, Sybase, Interbase, Firebird, FrontBase, Microsoft SQL Server, SAP DB,HypersonicSQL (HSQLDB),Progress.

2.7.2. Ventajas

Las ventajas son las siguientes:

- Productividad: evita mucho del código confuso de la capa de persistencia, permitiendo centrarse en la lógica de negocio.
- Mantenibilidad: por tener pocas líneas de código permite que el código sea más claro. Al dividir la capa de persistencia se puede identificar los errores muy fácilmente.
- Rendimiento: Hibernate tiene un buen desempeño pero todo depende realmente de cómo se realicen las consultas y cómo se configure el Framework.
- Indepencia del proveedor: una solución ORM te abstrae del SGBD. Permite desarrollar en local con bases de datos ligeras sin implicación en el entorno de producción.

2.7.3. Componentes básicos

Para utilizar Hibernate es importante saber que éste se compone de un conjunto de librerías reunidas en varios APIs, en donde, dependiendo del problema se debe escoger y configurar, las librerías necesarias en el proyecto. [19]

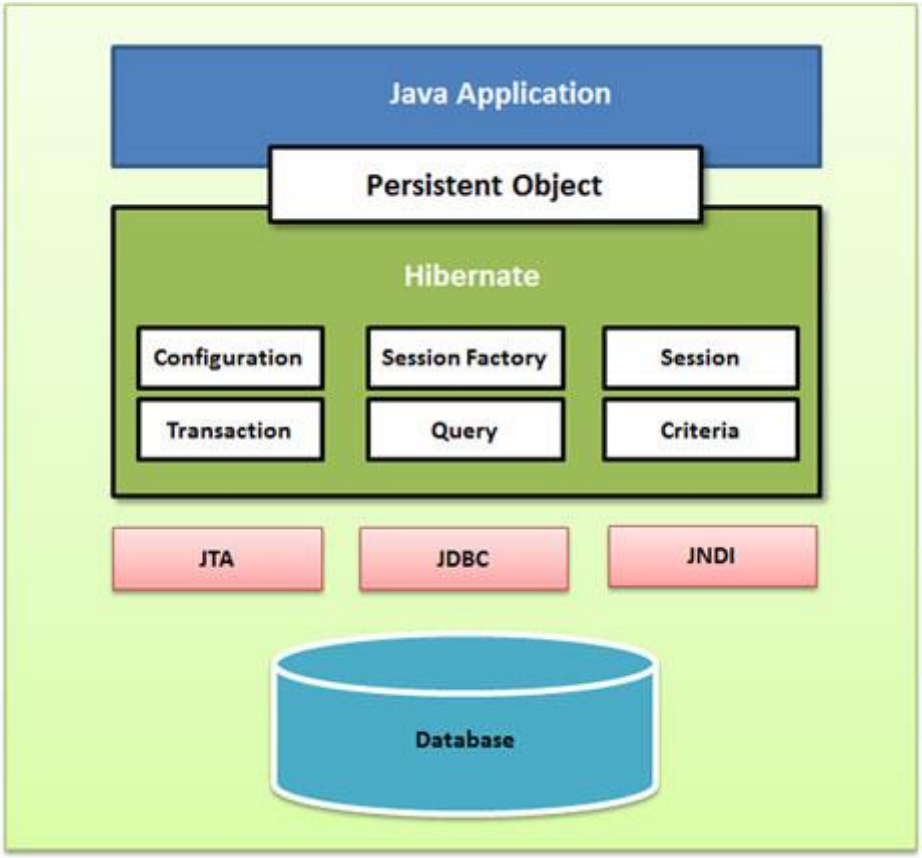

Figura 17. Componentes de Hibernate

De todo el conjunto de APIs de Hibernate, existen varias clases que permiten el trabajo básico con el Framework. Entre las más importantes se encuentran:

- Session: corresponde con un objeto que representa una unidad de trabajo con la base de datos (transacción). Además representa el gestor de persistencia, ya que dispone de la API básica para poder cargar y guardar objetos.
- Transaction: La API de Hibernate contiene utilidades para demarcar la transaccionalidad de operaciones de manera programática.
- Query: este interfaz permite crear consultas y enlazar argumentos a parámetros de la consulta (binding). Permite definir consultas en HQL (Hibernate Query Language) o en SQL.
- SessionFactory: es una factoría de sesiones. Proporciona objetos Session. Además, es *thread-safe* y permite concurrencia.

2.7.4. Conclusiones

El uso de aplicaciones de persistencia de objetos en bases de datos relacionales es cada vez más común. En el mercado existen muchos frameworks dedicados a esta tarea que hacen el trabajo de almacenamiento de objetos de una forma transparente para el programador.

Hibernate es uno de los mejores frameworks de persistencia de la actualidad ya que posee características mejoradas como: buen soporte, facilidad de adaptación, categoría libre, soporte de java y .net , buena documentación y un grupo de programadores mundiales que soportan su crecimiento.

Con Hibernate podremos cubrir de manera sencilla y rápida el 80 - 90% de la persistencia de nuestra aplicación. Esto nos permite centrar nuestros esfuerzos en optimizar las consultas que realmente lo merecen.

2.8. JPA

Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java.

JPA es una parte de la especificación de EJB 3.0 (JSR 220). Por lo tanto no existe realmente como framework, sino que es simplemente un documento. Un documento en el cual se especifican los principios básicos de gestión de la capa de persistencia en el mundo de Java EE. [20]

El uso de JPA no se limita a los componentes software EJB. Se puede utilizar en aplicaciones web y aplicaciones clientes. Para ello, combina ideas y conceptos de los principales frameworks de persistencia, como Hibernate, Toplink y JDO. El mapeo objeto-relacional (es decir, la relación entre entidades Java y tablas de la base de datos, queries con nombre, etc) se realiza mediante anotaciones en las propias clases de entidad.

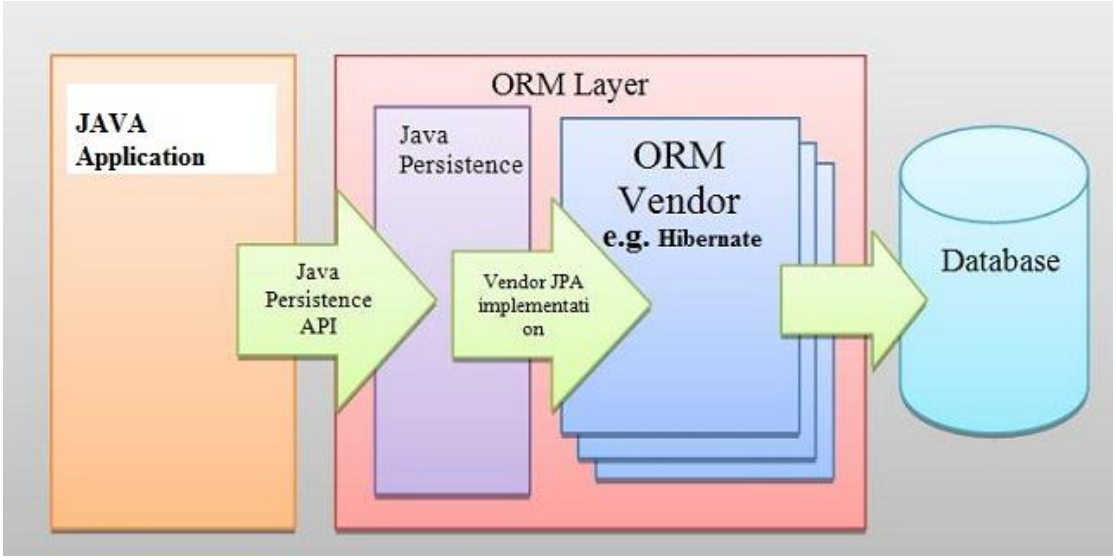

Figura 18. Ejemplo de uso de JPA

La relación que existe entre JPA e Hibernate es que este último implementa como parte de su código la especificación de JPA. Es decir podemos usar Hibernate para construir una capa de persistencia apoyándonos en las definiciones y reglas de la especificación de JPA, aunque no es obligatorio.

Eso sí, esto no quiere decir que Hibernate simplemente implemente el standard de JPA. Hibernate es mucho más grande que la especificación de JPA y añade más funcionalidad.

2.8.1. Persistencia

Pero para entender JPA, tendremos que tener claro el concepto "persistencia".
La persistencia o el almacenamiento permanente, es una de las necesidades básicas de cualquier sistema de información de cualquier tipo. En primer lugar, se propuso que los programas trataran los datos haciendo consultas directas a la base de datos. Después, se propuso trabajar con objetos, pero las bases de datos tradicionales no admiten esta opción.

Debido a esta situación, aparecieron los motores de persistencia, cuya función es traducir entre los dos formatos de datos: de registros a objetos y de objetos a registros. Persistir objetos Java en una base de datos relacional implica serializar un árbol de objetos Java en una base de datos de estructura tabular y viceversa. Esencial es la necesidad de mapear objetos Java para optimizar velocidad y eficiencia de la base de datos.

Una entidad pasará a ser manejada por el contexto de persistencia de JPA cuando ésta sea persistida (mediante el método persist del Entity Manager). En este caso, y mientras la entidad sea manejada/asociada por el contexto de persistencia (también se las conoce como attached entities), el estado (valores de la propiedades) de la entidad será automáticamente sincronizado con la BD.

La unidad de persistencia define un conjunto de todas las entidades que son gestionadas por la instancia del EntityManager en una aplicación. Este conjunto de clases de entidad representa los datos contenidos en una única BBDD.
Las unidades de persistencia se definen en el fichero de configuración persistence.xml.

2.8.2. Interfaces JPA

Las interfaces de las que se compone JPA son:

- javax.persistence.Persistence: contiene métodos estáticos de ayuda para obtener una instancia de Entity Manager Factory de una forma independiente al vendedor de la implementación de JPA. Una clase de inicialización que va a proporcionar un método estático para la creación de una Entity Manager Factory.
- javax.persistence.EntityManagerFactory: la clase EntityManagerFactory nos ayuda a crear objetos de EntityManager utilizando el patrón de diseño del Factory. Este objeto en tiempo de ejecución representa una unidad de

persistencia particular. Generalmente va a ser manejado como un singleton y proporciona métodos para la creación de instancias EntityManager.

- javax.persistence.Entity: la clase Entity es una anotación Java que se coloca a nivel de clases Java serializables y que cada objeto de una de estas clases anotadas representa un registro de una base de datos.
- javax.persistence.EntityManager: es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones. Cada Entity Manager puede realizar operaciones CRUD (Create, Read, Update, Delete) sobre un conjunto de objetos persistentes. Es un objeto único, no compartido que representa una unidad de trabajo particular para el acceso a datos. Proporciona métodos para gestionar el ciclo de vida de las instancias entidad y para crear instancias Query.
- javax.persistence.Query: la interface javax.persistence.Query está implementada por cada vendedor de JPA para encontrar objetos persistentes manejando cierto criterio de búsqueda. JPA estandariza el soporte para consultas utilizando Java Persistence Query Language (JPQL) y Structured Query Language (SQL). Podemos obtener una instancia de Query desde una instancia de un Entity Manager.
- javax.persistence.EntityTransaction: cada instancia de Entity Manager tiene una relación de uno a uno con una instancia de javax.persistence.EntityTransaction, permite operaciones sobre datos persistentes de manera que agrupados formen una unidad de trabajo transaccional, en el que todo el grupo sincroniza su estado de persistencia en la base de datos o todos fallan en el intento;en caso de fallo, la base de datos quedará con su estado original. Maneja el concepto de todos o ninguno para mantener la integridad de los datos.

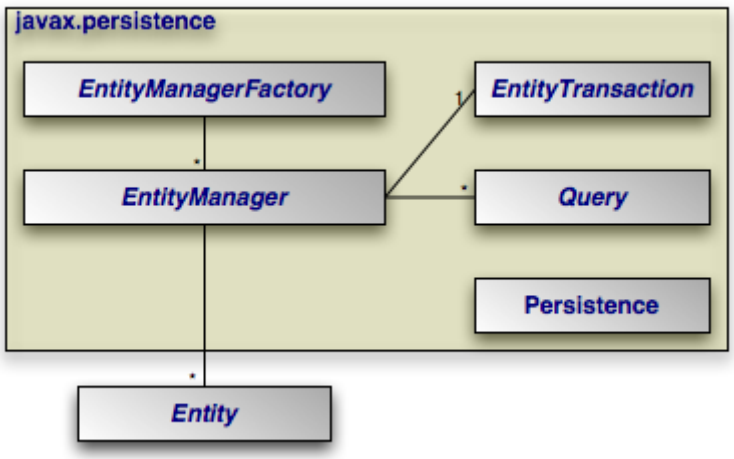

Figura 19. Interfaces de JPA

2.8.3. Anotaciones

JPA trabaja fuertemente con anotaciones. Gracias a las anotaciones, JPA mapea automáticamente nuestras clases en la base de datos de manera transparente, y utilizando un estándar, lo cual entre otras cosas nos permite poder migrar de motor cuando queramos sin ningún problema. [21]

Para mapear un bean (una clase java) con una tabla de la base de datos, tendríamos que escribir lo que se llama un Entity.

Esto es tan sencillo como escribir nuestro bean, con sus atributos y métodos get y set. Y después añadirle la anotación "@Entity" a la par que seleccionamos uno de sus atributos como clave primaria con "@Id". Por ejemplo, el siguiente código es el utilizado para crear la entidad "User".Mediante las anotaciones, JPA nos permitiría almacenar, recuperar, o actualizar campos sobre una tabla "users":

```
@Entity
@Table(name="users")
public class User implements Serializable {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;

    @NotEmpty
    @Column(nullable = false)
    private String nameUser;

}
```

Mediante @Entity indicamos que es una entidad. Con la anotación @Table forzamos a que se mapee con una tabla llamada "users".En caso de no usar la anotación @Table, el nombre de la tabla creada se llama exactamente igual al nombre de la clase java.

Con @Id vamos indicando qué columnas son clave y con @GeneratedValue podemos indicar si queremos que los campos clave se creen de manera automática, por ejemplo mediante el uso de secuencias. En otro caso, somos nosotros los que debemos calcular el valor de un nuevo id cuando vayamos a insertar, o podemos elegir IDENTITY y que se genere una clave autoincremental. Otra anotación es @Column, que nos permite asociar un atributo a un nombre de columna en particular. Con @NotEmpty indicamos que ese valor no puede estar vacío. [22]

Otra anotación interesante es @NamedQueries , con la que vamos indicando las diferentes sentencias SQL que queremos utilizar(en realidad sentencias JPQL).

También son importantes las anotaciones para indicar las relaciones de la entidad, las cuales veremos con más detalle a continuación.

2.8.4. Relaciones multiples de la entidad

Hay cuatro tipos de relaciones: uno a uno, uno a muchos, muchos a uno, y muchos a muchos.

- Uno a uno: cada entidad se relaciona con una sola instancia de otra entidad. Las relaciones uno a uno utilizan anotaciones de la persistencia de java "OneToOne". Ej:@OneToOne(cascade={CascadeType.*PERSIST*, CascadeType.*REMOVE*})
- Uno a muchos: una entidad puede estar relacionada con varias instancias de otras entidades. Las relaciones uno a muchos utilizan anotaciones de la persistencia de java "OneToMany" en los campos o propiedades persistentes. Ej:@OneToMany(cascade=CascadeType.*ALL*, fetch=FetchType.*EAGER*, mappedBy="group").
- Muchos a uno: múltiples instancias de una entidad pueden estar relacionadas con una sola instancia de otra entidad. Esta multiplicidad es lo contrario a la relación uno a muchos. Las relaciones muchos a uno utilizan anotaciones de la persistencia de java "ManyToOne" en los campos o propiedades persistentes. Ej:@ManyToOne
- Muchos a muchos: en este caso varias instancias de una entidad pueden relacionarse con múltiples instancias de otras entidades. Este tipo de relación utiliza anotaciones de la persistencia de java "ManyToMany" en los campos o propiedades persistentes.
  Ej:@ManyToMany
  @JoinTable(name="groups_users",joinColumns=@JoinColumn(name="group_id"),inverseJoinColumns=@JoinColumn(name="user_id"))

Capítulo 3

Análisis

3.1. Introducción

Una vez estudiado las tecnologías pertinentes, es el momento de realizar la aplicación web que represente en cierta medida a estas tecnologías.

En primer lugar, se especifican los requisitos de usuario, que se dividen en dos categorías:

- Requisitos de usuario de capacidad: indican qué funciones puede o tiene que hacer la aplicación para cumplir los propósitos requeridos para el sistema.
- Requisitos de usuario de restricción: indican qué limitaciones tiene la aplicación.

A continuación, se definirán los roles o tipos de usuario que pueden utilizar la aplicación y las necesidades tecnológicas del sistema desde el punto de vista de los usuarios.

Por último, se tratarán los requisitos desde el punto de vista del diseño. Estos requisitos se han dividido en dos tipos:

- Requisitos funcionales: son los procesos que se pueden realizar en la plataforma como por ejemplo el registro de usuarios, eventos, autenticaciones…
- Requisitos no funcionales: Son condiciones que debe cumplir el proyecto como pueden ser la rapidez, la seguridad y el coste.

Por lo tanto, en este capítulo se analizarán los requisitos necesarios para cumplir los objetivos planteados y se detallarán los roles que existirán dentro de la aplicación web.

3.2. Requisitos de usuario

Para facilitar la lectura, los requisitos de usuario serán recogidos en tablas cuyo formato incluirá los siguientes campos:

- Nombre del requisito : nombre identificativo del requisito.
- Identidicador: es el código único que identifica a cada requisito. Su nomenclatura ha de seguir el formato: RU_ XY_ZZ, donde:
    - XY: es UC si es un requisito de usuario de capacidad y UR si es un requisito de usuario de restricción.
    - ZZ : será sustituido por el número de requisito dentro de su categoría.
- Descripción: descripción detallada del requisito correspondiente.

- Necesidad: indica la importancia del requisito desde el punto de vista del usuario. Los valores pueden ser Esencial y Conveniente.
- Prioridad: indica la prioridad de implementación del requisito. Los valores pueden ser *Baja*, *Media* y *Alta*.
- Estabilidad: indica el grado de variacion que puede experimentar el requisito a lo largo del desarrollo del proyecto. Los valores pueden ser Estable o Inestable.
- Verificabilidad: indica la dificultad que posee la verificacion del cumplimiento del requisito. Los valores pueden ser Baja, Media o Alta.

3.2.1 Requisitos de capacidad

A continuación, se detallan los requisitos de capacidad para la aplicación:

REGISTRO DE USUARIO	
IDENTIFICADOR: RU_UC_01	
DESCRIPCIÓN: Para poder utilizar los servicios de la aplicación, el usuario debe estar registrado.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

MODIFICACIÓN DE PERFIL DE USUARIO	
IDENTIFICADOR: RU_UC_02	
DESCRIPCIÓN: Un usuario puede acceder a la información de sus datos personales y modificarlos si lo considera oportuno.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMÁGEN DE PERFIL DE USUARIO	
IDENTIFICADOR: RU_UC_03	
DESCRIPCIÓN: Un usuario puede tener una imagen asociada a su perfil si lo desea. Si a la hora de registrarse, no selecciona ninguna el sistema debe asociar una imagen por defecto.	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

AUTENTICACIÓN DE USUARIO	
IDENTIFICADOR: RU_UC_04	
DESCRIPCIÓN: Si un usuario ya está registrado, puede acceder a los servicios de la aplicación introduciendo su nombre de usuario y password.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

CREAR GRUPO	
IDENTIFICADOR: RU_UC_05	
DESCRIPCIÓN: Cualquier usuario puede crear un grupo y añadir usuarios a ese grupo. Cualquier evento que sea creado en dicho grupo, será notificado a los usuarios que pertenecen al grupo	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Inestable	**VERIFICABILIDAD:** Alta

MODIFICAR GRUPOS	
IDENTIFICADOR: RU_UC_06	
DESCRIPCIÓN: Se podrán modificar datos del grupo, añadir nuevos usuarios al grupo o eliminar usuarios del grupo. En caso de eliminar el grupo, todos los eventos del grupo serán borrados automáticamente.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

CREAR REGIONES	
IDENTIFICADOR: RU_UC_07	
DESCRIPCIÓN: Cuando se crea una región, automáticamente se crea un grupo asociado a dicha región. Los grupos asociados a las regiones son los grupos por defecto que un usuario puede elegir como grupo inicial en el proceso de registrarse.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

MODIFICAR REGIONES	
IDENTIFICADOR: RU_UC_08	
DESCRIPCIÓN: Al ser modificado algún nombre de los datos de la región, automáticamente se modificará el nombre del grupo asociado .En caso de borrar una región, el grupo asociado a la región automáticamente será borrado.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

CREAR EVENTO	
IDENTIFICADOR: RU_UC_09	
DESCRIPCIÓN: Cualquier usuario puede crear un evento, indicando descripción, fecha, lugar y otros datos de interés del evento si son necesarios. El grupo al que pertenece el evento se elegirá entre los grupos a los que pertenece el usuario.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

MODIFICAR EVENTO	
IDENTIFICADOR: RU_UC_10	
DESCRIPCIÓN: Se podrá modificar algún dato del evento, como el lugar, la fecha, la descripción, etc. o borrar dicho evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

BUSCAR USUARIOS	
IDENTIFICADOR: RU_UC_11	
DESCRIPCIÓN: Cualquier usuario podrá acceder a una lista de todos los usuarios registrados de la aplicación y hacer búsquedas por nombre de usuario	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

BUSCAR GRUPOS	
IDENTIFICADOR: RU_UC_12	
DESCRIPCIÓN: Cualquier usuario podrá acceder a una lista de todos los grupos registrados de la aplicación y hacer búsquedas por nombre de grupo.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

BUSCAR EVENTOS	
IDENTIFICADOR: RU_UC_13	
DESCRIPCIÓN: Cualquier usuario podrá acceder a una lista de todos los eventos registrados de la aplicación y hacer búsquedas por nombre de evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

BUSCAR REGIONES	
IDENTIFICADOR: RU_UC_14	
DESCRIPCIÓN: El administrador podrá acceder a una lista de todas las regiones registradas en la aplicación y hacer búsquedas por nombre de región.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Inestable	**VERIFICABILIDAD:** Alta

LISTAR MIS EVENTOS	
IDENTIFICADOR: RU_UC_15	
DESCRIPCIÓN: Un usuario puede acceder a una lista de todos los eventos en los que participa y hacer búsquedas por nombre de evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

LISTAR MIS GRUPOS	
IDENTIFICADOR: RU_UC_16	
DESCRIPCIÓN: Un usuario puede acceder a una lista de todos los grupos a los que pertenece y hacer búsquedas por nombre de grupo.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

NOTIFICAR ÚLTIMOS CAMBIOS	
IDENTIFICADOR: RU_UC_17	
DESCRIPCIÓN: Al acceder un usuario a la aplicación será informado de todos los nuevos eventos o modificaciones en los eventos a los que está apuntado que han ocurrido desde la última vez que accedió a la aplicación.El usuario podrá ver un listado de dichos eventos	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Inestable	**VERIFICABILIDAD:** Alta

UNIRSE A UN GRUPO/ABANDONAR UN GRUPO	
IDENTIFICADOR: RU_UC_18	
DESCRIPCIÓN: Un usuario podrá unirse a un grupo de los que hay registrados en el sistema. Del mismo modo, podrá abandonar cualquier grupo en el momento que lo necesite.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

UNIRSE A UN EVENTO/ABANDONAR UN EVENTO	
IDENTIFICADOR: RU_UC_19	
DESCRIPCIÓN: Un usuario podrá unirse a cualquier evento de los grupos a los que pertenece el usuario. Del mismo modo, podrá abandonar un evento en el momento que lo necesite.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

LISTAR EVENTOS DE MIS GRUPOS	
IDENTIFICADOR: RU_UC_20	
DESCRIPCIÓN: Un usuario puede acceder a una lista de todos los eventos de los grupos a los que pertenece.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

EVENTOS ORDENADOS POR FECHA	
IDENTIFICADOR: RU_UC_21	
DESCRIPCIÓN: Los listados de eventos estarán ordenados según la fecha del evento, colocándose en primer lugar los eventos cuya fecha de suceso son más próximos.	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

EVENTO YA REALIZADO	
IDENTIFICADOR: RU_UC_22	
DESCRIPCIÓN: Los eventos cuya fecha y hora de suceso son anteriores a la actual, permanecerán registrados en el sistema. El administrador de cada evento será el responsable de borrarlos del sistema si lo considera oportuno. Los eventos ya realizados sólo se mostrarán en el listado de “mis eventos”.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

3.2.2. Requisitos de restricción

Los requisitos de restricción son los siguientes:

IMPOSIBILIDAD DE CAMBIAR GRUPO DEL EVENTO	
IDENTIFICADOR: RU_UR_01	
DESCRIPCIÓN: No se podrá modificar el grupo al que pertenece el evento una vez que ha sido creado dicho evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

TAMAÑO DE IMAGEN DE PERFIL LIMITADO	
IDENTIFICADOR: RU_UR_02	
DESCRIPCIÓN: La imagen que puede asociar el usuario a su perfil no debe ocupar mucho espacio en base de datos.	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE CREAR/EDITAR REGIONES	
IDENTIFICADOR: RU_UR_03	
DESCRIPCIÓN: Ningún usuario puede crear, modificar o borrar regiones. El usuario no puede acceder a ningún dato relativo con las regiones. Sólo el administrador podrá realizar las acciones anteriores.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE UNIRSE A UN EVENTO	
IDENTIFICADOR : RU_UR_04	
DESCRIPCIÓN: Un usuario no podrá unirse a un evento si el usuario no pertenece al grupo del evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE EDITAR USUARIOS	
IDENTIFICADOR: RU_UR_05	
DESCRIPCIÓN: Un usuario sólo puede modificar su perfil o borrarlo si es el propio usuario.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE EDITAR GRUPOS	
IDENTIFICADOR: RU_UR_06	
DESCRIPCIÓN: Un usuario no puede modificar información de un grupo, ni añadir usuarios a un grupo, ni eliminar usuarios del grupo, ni eliminar dicho grupo si no es el administrador del grupo.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE EDITAR EVENTOS	
IDENTIFICADOR: RU_UR_07	
DESCRIPCIÓN: Un usuario no puede modificar información de un evento como cambiar el lugar, la fecha, la hora, la descripción, etc. si el usuario no es el administrador del evento.	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

3.3. Características de los usuarios

Se han definido dos roles o usuarios diferentes que pueden utilizar la aplicación. Se diferencian básicamente por los permisos que poseen y de las acciones que pueden llegar a desempeñar dentro de la aplicación web.

- Administrador: gestiona y mantiene la aplicación en constante funcionamiento. Gestiona todos los datos que componen el conjunto de información manejado por el sistema, por lo que tiene total libertad para administrar regiones, usuarios, grupos y eventos. Desempeña también una labor de desarrollador, puesto que es el encargado de añadir nuevas funcionalidades a la aplicación.

- Usuario: un usuario registrado tiene acceso a poder crear grupos y eventos. Puede a su vez administrar los grupos y eventos que han sido creados por el mismo, y también puede modificar sus propios datos personales.

Se entiende que una misma persona puede poseer varios roles. No se tienen en cuenta los usuarios indirectos de la aplicación, que no acceden a ella, como son los gestores de las bases de datos o los administradores de red que se encargan del buen funcionamiento de la misma. Estas acciones recaen directamente en el usuario administrador.

3.4. Entorno operacional

En este punto se analizan las necesidades tecnológicas del sistema desde el punto de vista de los usuarios. Se debe distinguir entre las necesidades de cada uno de los roles anteriormente comentadas:

ADMINISTRADOR
Ordenador con acceso a redes. Requiere los elementos para poder desarrollar las herramientas en el lenguaje deseado, así como poder añadir funcionalidades a la aplicación. Por lo tanto, se necesita un equipo con Eclipse o algún otro IDE similar y con MySQL como sistema gestor de base de datos.

Tabla 1. Entorno operacional del administrador

USUARIO
Cualquier ordenador que contenga un acceso a la red y la máquina virtual Java.

Tabla 2. Entorno operacional del usuario

3.5. Casos de uso

Los personajes o entidades que participarán en un caso de uso se denominan actores. Un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

A continuación, vamos a describir los casos de uso desarrollados para la aplicación web, especificando los prerrequisitos para llevarlos a cabo, los posibles escenarios secundarios y su descripción detallada.

Para facilitar la lectura, los casos de uso se muestran en tablas cuyo formato incluirá los siguientes campos:

- Identificador: código único que identifica a cada caso de uso. Se compone del siguiente formato:
    - CU: corresponde a *Caso Uso*.
    - XX: número del caso de uso.
- Nombre: nombre del caso de uso.Este nombre se usa en el diagrama de casos de uso para identificarlo.
- Actores: agentes externos que interactúan con el sistema.
- Precondiciones: condiciones necesarios para que el caso de uso se pueda realizar.
- Postcondiciones: hechos que suceden si el caso de uso se ha ejecutado correctamente.
- Descripción: descripción detallada del caso de uso correspondiente.
- Escenario principal: descripción detallada de los pasos que ha de realizar el actor dentro del escenario para completar el caso de uso.
- Escenario secundario: descripción detallada de los pasos que ha de realizar el actor dentro del escenario, en caso de que pueda darse una bifurcación en la ejecución normal del sistema.

Los casos de uso son los siguientes:

UC1 : REGISTRAR USUARIO
ACTORES: Usuario.
PRECONDICIONES: Ninguna.
POSTCONDICIONES: El usuario estará registrado en el sistema y podrá utilizar los servicios.
DESCRIPCIÓN: Para que un usuario pueda utilizar los servicios de la aplicación web, necesita estar registrado.
ESCENARIO PRINCIPAL: 1. El usuario accede a la página principal de la aplicación web, y pulsa el botón "Registro". 2. Aparece un formulario de registro, y el usuario debe introducir una serie de datos personales, entre ellos alguno obligatorio como nombre de usuario y password, y elegir el grupo principal al que va a pertenecer. También tiene la opción de asociar una foto a su perfil. 3. Al pulsar el botón "Registrar" el formulario será enviado, procesado y validado. Una vez validado, se muestra el menú principal.
ESCENARIO SECUNDARIO: 3. Al pulsar el botón "Registrar" la aplicación muestra de nuevo la página del formulario de registro indicando un mensaje de que el nombre de usuario ya existe.

UC2 : AUTENTICAR USUARIO
ACTORES: Administrador, Usuario.
PRECONDICIONES : El usuario debe estar registrado en la base de datos de la aplicación
POSTCONDICIONES: El usuario podrá utilizar los servicios de la aplicación.
DESCRIPCIÓN: El usuario necesita ser autenticado para poder acceder a ciertas características de la aplicación.
ESCENARIO PRINCIPAL: 1. El usuario accede a la página principal de la aplicación web, y pulsa el botón "Login". 2. Aparece un formulario de autenticación, y el usuario debe rellenar como datos obligatorios tanto el nombre de usuario como el password. 3. Al pulsar el botón "Login" el usuario es autenticado y se muestra la pantalla de menú principal.
ESCENARIO SECUNDARIO: 3. Al pulsar el botón "Login" la aplicación muestra de nuevo la página del formulario de autenticación indicando un mensaje de que el usuario y/o el password no son válidos.

UC3: CREAR REGIÓN
ACTORES: Administrador.
PRECONDICIONES: El administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: Se crea una región y un grupo asociado a la región.
DESCRIPCIÓN: Las regiones tienen asociados grupos que son los que aparecen como posibles grupos iniciales al registrarse un usuario.
ESCENARIO PRINCIPAL: 1. El administrador pulsa el botón “REGIONES” colocado en el menú de botones lateral. 2. En la nueva ventana, se pulsa en “CREAR REGIÓN”. 3. En el formulario de crear región, se rellenan los campos de Nombre de Región y de País. Ambos campos son obligatorios. 4. Se pulsa el botón “Crear” y la región es creada y también un grupo asociado a la región cuyo nombre depende de lo que hayamos rellenado en el formulario del paso 3.
ESCENARIO SECUNDARIO: 4. Se pulsa el botón “Crear” y se vuelve al formulario de crear región mostrando un mensaje de que ya existe un grupo con ese nombre.

UC4 : MODIFICAR REGIÓN
ACTORES: Administrador.
PRECONDICIONES: El administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: Se modifica la región y el grupo asociado a la región.
DESCRIPCIÓN: La modificación consiste en cambiar o el nombre de la región o el del país.
ESCENARIO PRINCIPAL: 1. El administrador pulsa el botón “REGIONES” colocado en el menú de botones lateral. 2. Aparece una lista de las regiones. El administrador pulsa en el enlace “modificar” de la región que desea editar. 3. Se muestra un formulario con los datos de la región. El administrador puede cambiar el nombre de la región o del país, pero no puede dejar vacío ninguno de los campos. 4. Una vez realizado el cambio deseado, se pulsa en “Editar Región”.
ESCENARIO SECUNDARIO: 4. Una vez realizado el cambio deseado, se pulsa en “Editar Región”. Se vuelve a mostrar el formulario de editar región con un mensaje indicando que no se ha podido modificar porque ya existe un grupo con ese nombre.

UC5: CREAR EVENTO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario/administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: Se crea el evento. El evento es añadido al grupo al que pertenece. Se envía una notificación a los usuarios del grupo.
DESCRIPCIÓN: Un evento es creado para que otros usuarios puedan unirse al mismo. A la hora de crear el evento hay que indicar cierta información como lugar, hora, fecha, etc.
ESCENARIO PRINCIPAL: 1. El usuario pulsa el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa en el botón “CREAR EVENTO”. 3. Aparece el formulario de crear evento. El usuario podrá indicar la localización con un marcador en un mapa de Google Maps, indicar la hora y fecha del evento y otros datos como el título y la descripción del evento por ejemplo. 4. Se pulsa el botón “Crear Evento” y el evento es creado.
ESCENARIO SECUNDARIO: 3. El usuario rellena el formulario, indicando hora, fecha y descripción. 4. Se pulsa el botón “Crear Evento” y se muestra de nuevo el formulario de crear un evento y un mensaje indicando que el título del evento debe ser rellenado.

UC6: MODIFICAR EVENTO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario/administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: El evento es modificado. Se envía una notificación a los usuarios unidos al evento.
DESCRIPCIÓN: Un evento es modificado y se cambia alguno de los datos del mismo, como la localización, la fecha, la hora, etc.
ESCENARIO PRINCIPAL: 1. El usuario pulsa el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa en el botón “LISTAR EVENTOS”. 3. Aparece una lista de los eventos registrados .El usuario pincha el enlace “modificar” del evento que desea editar. 4. Se muestra la pantalla de editar evento. El usuario modifica algún dato del evento. 5. El usuario pulsa el botón “Modificar Evento” y el evento es modificado.

UC7: MODIFICAR USUARIO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El perfil que se pretende modificar debe ser el del propio usuario.
POSTCONDICIONES: El perfil es modificado.
DESCRIPCIÓN: Un usuario puede modificar sus datos personales o cambiar la foto asociada a su perfil.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en su nombre de usuario que se encuentra en la cabecera. 2. Se muestra el perfil del usuario. Se pulsa en el botón “Modificar Perfil”. 3. Aparece el formulario de modificar perfil de usuario. El usuario modifica los datos necesarios o cambia la foto de perfil. 4. Se pulsa el botón “Modificar Perfil” y el perfil es modificado.
ESCENARIO SECUNDARIO: 3. Aparece el formulario de modificar perfil. El usuario elige una foto de perfil de su ordenador y pulsa el botón “Modificar Perfil”. 4. Se vuelve a la pantalla de modificar perfil y se muestra un mensaje de error indicando que el tamaño del fichero que se había seleccionado es superior al límite permitido.

UC8: CREAR GRUPO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario/administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: El grupo es creado.
DESCRIPCIÓN: Se crea un grupo y se añaden usuarios al grupo para que puedan ser informados de todos los eventos que se crean en dicho grupo.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “CREAR GRUPO”. 3. Aparece el formulario de crear grupo. El usuario puede ir añadiendo usuarios al grupo. Cuando se está escribiendo en la caja de texto para añadir un usuario el sistema se encarga de buscar usuarios que contengan el patrón escrito. 4. Una vez añadidos los usuarios se pulsa en el botón “Crear Grupo” y el grupo es creado.
ESCENARIO SECUNDARIO: 3. Aparece el formulario de crear grupo. El usuario añade usuarios al grupo pero no introduce el nombre del grupo. 4. El usuario pulsa el botón “Crear Grupo” y se muestra de nuevo la pantalla de crear grupo con un mensaje de error indicando que el nombre del grupo debe ser rellenado.

UC9: MODIFICAR GRUPO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe ser el administrador del grupo para poder modificarlo.
POSTCONDICIONES: El grupo es modificado.
DESCRIPCIÓN: Se modifica el grupo añadiendo nuevos usuarios o eliminando usuarios del grupo.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR GRUPOS”. 3. Aparece una lista de los grupos registrados .El usuario pincha el enlace “modificar” del grupo que desea editar. 4. Se muestra la pantalla de modificar grupo con los usuarios que pertenecen al grupo. El usuario puede añadir nuevos usuarios o eliminar usuarios del grupo. 5. El usuario pulsa el botón “Modificar Grupo” y el grupo es editado.
ESCENARIO SECUNDARIO: 4. Se muestra la pantalla de modificar grupo. El usuario añade un usuario que ya existía en el grupo. Se muestra una ventana de alerta indicando al usuario que no se puede añadir al grupo.

UC10: BORRAR GRUPO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe ser el administrador del grupo para poder borrarlo.
POSTCONDICIONES: El grupo es borrado y todos los eventos del grupo también son borrados.
DESCRIPCIÓN: Un grupo es eliminado y todos sus eventos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR GRUPOS”. 3. Aparece una lista de los grupos registrados .El usuario pincha el enlace “borrar grupo” del grupo que desea eliminar.

UC11: BORRAR EVENTO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe ser el administrador del evento para poder borrarlo.
POSTCONDICIONES: El evento es borrado.
DESCRIPCIÓN: Un evento es eliminado y deja de estar registrado en el sistema.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR EVENTOS”. 3. Aparece una lista de los eventos registrados .El usuario pincha el enlace “borrar evento” del evento que desea eliminar.

UC12: BORRAR USUARIO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar registrado en el sistema.
POSTCONDICIONES: El usuario es borrado.
DESCRIPCIÓN: Un usuario es eliminado y deja de estar registrado en el sistema.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “USUARIOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR USUARIOS”. 3. Aparece una lista de los usuarios registrados .El usuario introduce su nombre de usuario en la caja de texto de búsqueda y pulsa el botón “Buscar”. 4. En la lista sólo debería salir el mismo usuario. Para eliminar el usuario hay que pulsar el enlace “borrar usuario”.

UC13: BORRAR REGIÓN
ACTORES: Administrador.
PRECONDICIONES: El administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: La región es borrada, junto con el grupo asociado y los eventos que pertenecen a dicho grupo.
DESCRIPCIÓN: Una región es eliminada y deja de estar registrada en el sistema.
ESCENARIO PRINCIPAL: 1. El administrador pulsa en el botón “REGIONES” colocado en el menú de botones lateral. 2. Aparece una lista de las regiones registradas. Para eliminar una región hay que pulsar el enlace “borrar región”.

UC14: LISTAR REGIONES
ACTORES: Administrador.
PRECONDICIONES: El administrador debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todas o varias regiones que están registradas en el sistema.
DESCRIPCIÓN: Se muestra un listado de las regiones y un conjunto de acciones a realizar con cada una de ellas.
ESCENARIO PRINCIPAL: 1. El administrador pulsa en el botón “REGIONES” colocado en el menú de botones lateral. 2. Se muestra un listado de las regiones registradas en el sistema junto con un conjunto de acciones a realizar para cada una de ellas como puede ser modificarla o eliminarla. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.

UC15: LISTAR USUARIOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todos o varios usuarios que están registrados en el sistema.
DESCRIPCIÓN: Se muestra un listado de los usuarios y un conjunto de acciones a realizar con cada uno de ellos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “USUARIOS” colocado en el menú de botones lateral. 2. Se muestra un listado de los usuarios registrados en el sistema junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver el perfil del usuario, modificarlo o eliminarlo. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.

UC16: LISTAR TODOS LOS GRUPOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todos o varios grupos que están registrados en el sistema.
DESCRIPCIÓN: Se muestra un listado de los grupos y un conjunto de acciones a realizar con cada uno de ellos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón "GRUPOS" colocado en el menú de botones lateral. 2. El usuario pulsa el botón "LISTAR GRUPOS". 3. Se muestra un listado de los grupos registrados en el sistema junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver la información del grupo, modificarlo, eliminarlo, abandonar el grupo o unirse al grupo dependiendo de si formábamos parte del grupo o no. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.

UC17: LISTAR TODOS LOS EVENTOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todos o varios eventos que están registrados en el sistema.
DESCRIPCIÓN: Se muestra un listado de los eventos y un conjunto de acciones a realizar con cada uno de ellos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón "EVENTOS" colocado en el menú de botones lateral. 2. El usuario pulsa el botón "LISTAR EVENTOS". 3. Se muestra un listado de los eventos registrados en el sistema junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver la información del evento, modificarlo, borrarlo, abandonar el evento o unirse al evento. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.
ESCENARIO SECUNDARIO: 3. No hay ningún evento registrado y se muestra en pantalla el mensaje "EL LISTADO NO CONTIENE NINGÚN EVENTO"

UC18: LISTAR MIS EVENTOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todos o varios eventos en los que participa el usuario.
DESCRIPCIÓN: Se muestra un listado de los eventos en los que participa el usuario y un conjunto de acciones a realizar con cada uno de ellos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “MIS EVENTOS”. 3. Se muestra un listado de los eventos en los que participa el usuario junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver la información del evento, modificarlo o borrarlo si el usuario es administrador del mismo, o abandonar el evento. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.
ESCENARIO SECUNDARIO: 3. No hay ningún evento en el que participa el usuario y se muestra en pantalla el mensaje “EL LISTADO NO CONTIENE NINGÚN EVENTO”

UC19: LISTAR MIS GRUPOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Se muestran todos o varios grupos a los que pertenece el usuario.
DESCRIPCIÓN: Se muestra un listado de los grupos y un conjunto de acciones a realizar con cada uno de ellos.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR MIS GRUPOS”. 3. Se muestra un listado de los grupos a los que pertenece el usuario junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver la información del grupo, modificarlo o borrarlo si el usuario es administrador del mismo, o abandonar el grupo. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado.

UC20: UNIRSE A UN EVENTO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Todas las modificaciones que sucedan en el evento serán notificados al usuario.
DESCRIPCIÓN: Un usuario visualiza información de los eventos disponibles y decide unirse a un evento específico.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR EVENTOS”. 3. Se muestra un listado de todos los eventos. El usuario pulsa en el enlace “unirse al evento” del evento del que quiere participar.
ESCENARIO SECUNDARIO: 3. Se muestra un listado de todos los eventos. El usuario pulsa en el enlace “ver” del evento del que quiere visualizar información. 4. El usuario visualiza la información del evento. Para unirse al evento el usuario pulsa en el botón “Unirse al evento”

UC21: ABANDONAR UN EVENTO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación y participar en el evento que desea abandonar.
POSTCONDICIONES: El usuario dejará de recibir información del evento.
DESCRIPCIÓN: Un usuario visualiza información de los eventos y decide abandonar uno específico.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “MIS EVENTOS”. 3. Se muestra un listado de los eventos en los que participa el usuario. El usuario pulsa en el enlace “abandonar evento”.
ESCENARIO SECUNDARIO: 3. Se muestra un listado de los eventos en los que participa el usuario. El usuario pulsa en el enlace “ver” del evento del que quiere visualizar información. 4. El usuario visualiza la información del evento. Para abandonar el evento el usuario pulsa en el botón “Abandonar evento”.

UC22: UNIRSE A UN GRUPO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: El usuario pertenecerá al grupo, y recibirá notificaciones sobre los eventos del grupo.
DESCRIPCIÓN: Un usuario visualiza información de los grupos y decide unirse a uno específico.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR GRUPOS”. 3. Se muestra un listado de los grupos registrados en el sistema. El usuario pulsa en el enlace “unirse al grupo”.
ESCENARIO SECUNDARIO: 3. Se muestra un listado de los grupos registrados. El usuario pulsa en el enlace “ver” del grupo del que quiere visualizar información. 4. El usuario visualiza la información del grupo. Para unirse al grupo el usuario pulsa en el botón “Unirse al grupo”.

UC23: ABANDONAR UN GRUPO
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación y pertenecer al grupo que desea abandonar.
POSTCONDICIONES: El usuario dejará de pertenecer al grupo.
DESCRIPCIÓN: Un usuario visualiza información de los grupos y decide abandonar uno específico.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “GRUPOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “MIS GRUPOS”. 3. Se muestra un listado de los grupos a los que pertenece el usuario. El usuario pulsa en el enlace “abandonar grupo”.
ESCENARIO SECUNDARIO: 3. Se muestra un listado de los grupos a los que pertenece el usuario. El usuario pulsa en el enlace “ver” del grupo del que quiere visualizar información. 4. El usuario visualiza la información del grupo. Para abandonar el grupo el usuario pulsa en el botón “Abandonar grupo”.

UC24: VISUALIZAR NOVEDADES DE EVENTOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Ninguna
DESCRIPCIÓN: Un usuario visualiza información sobre nuevos eventos, nuevos grupos o modificaciones de eventos a los que está apuntado.
ESCENARIO PRINCIPAL: 1. El usuario al autenticarse accede al menú principal. Se muestra un mensaje indicando que hay novedades en relación a nuevos eventos o modificaciones de eventos en los que el usuario está apuntado. 2. El usuario pulsa en el enlace disponible en el mensaje indicado en el punto 1. 3. Se muestra un listado de los cambios referente a los eventos.
ESCENARIO SECUNDARIO: 2. El usuario pulsa en el botón “NOVEDADES” situado en el menú de botones lateral. 3. Se muestra un listado de los cambios referente a los eventos.

UC25: LISTAR EVENTOS DE MIS GRUPOS
ACTORES: Administrador, Usuario.
PRECONDICIONES: El usuario debe estar autenticado en la aplicación.
POSTCONDICIONES: Ninguna
DESCRIPCIÓN: Se muestra un listado de los eventos que pertenecen a los grupos del usuario.
ESCENARIO PRINCIPAL: 1. El usuario pulsa en el botón “EVENTOS” colocado en el menú de botones lateral. 2. El usuario pulsa el botón “LISTAR EVENTOS DE MIS GRUPOS”. 3. Se muestra un listado de los eventos de los grupos a los que pertenece el usuario junto con un conjunto de acciones a realizar para cada uno de ellos como puede ser ver la información del evento, modificarlo o borrarlo si el usuario es administrador del mismo, o unirse al evento o abandonarlo. También puede hacer búsquedas específicas introduciendo un patrón de búsqueda en la caja de texto que se muestra encima del listado

3.6. Diagramas de casos de uso

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Los diagramas de casos de uso de la aplicación son los siguientes:

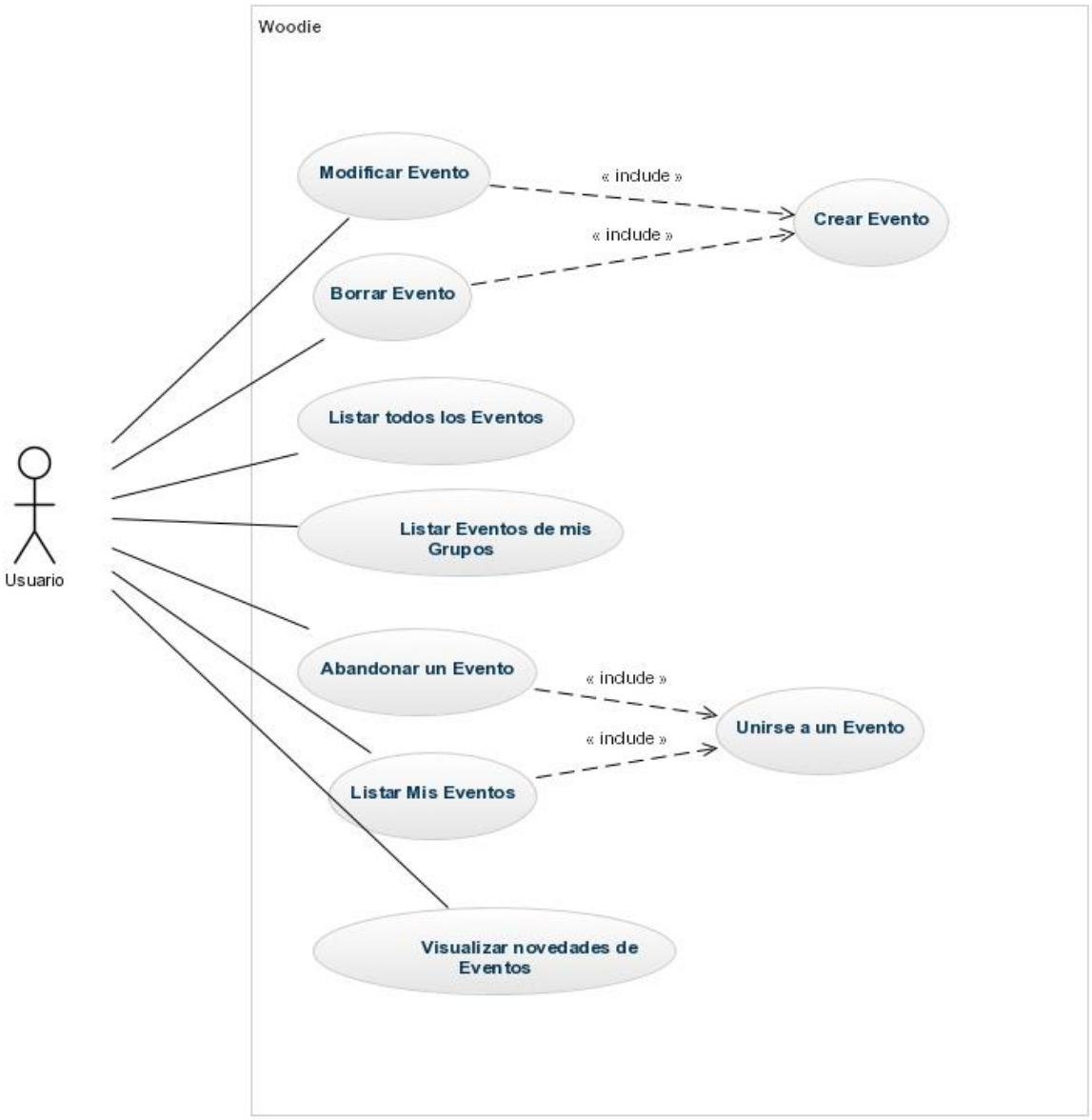

Figura 20. Diagrama de casos de uso del Usuario - Eventos

Figura 21. Diagrama de casos de uso del Usuario – Grupos y Usuarios

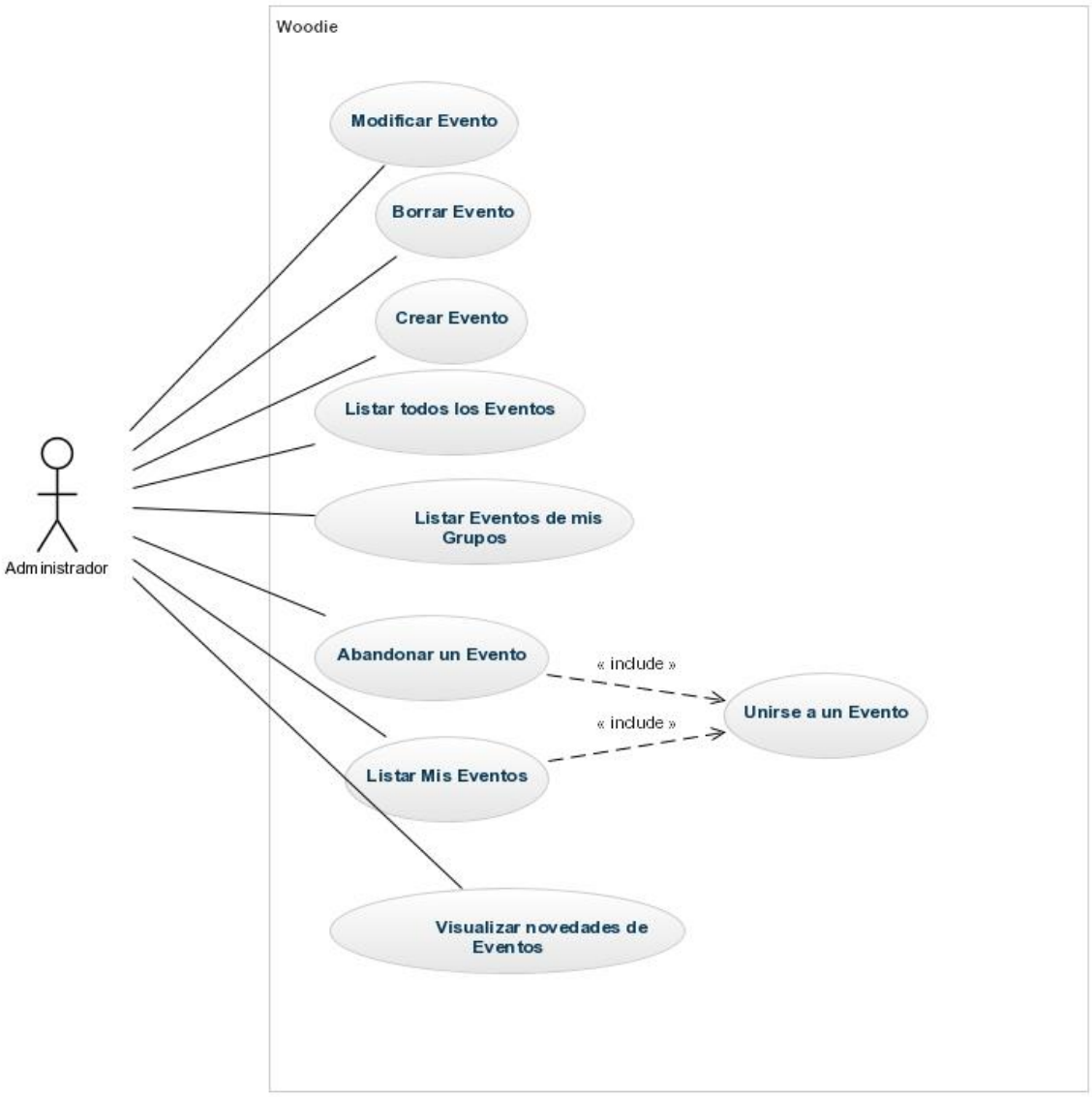

Figura 22. Diagrama de casos de uso del Administrador - Eventos

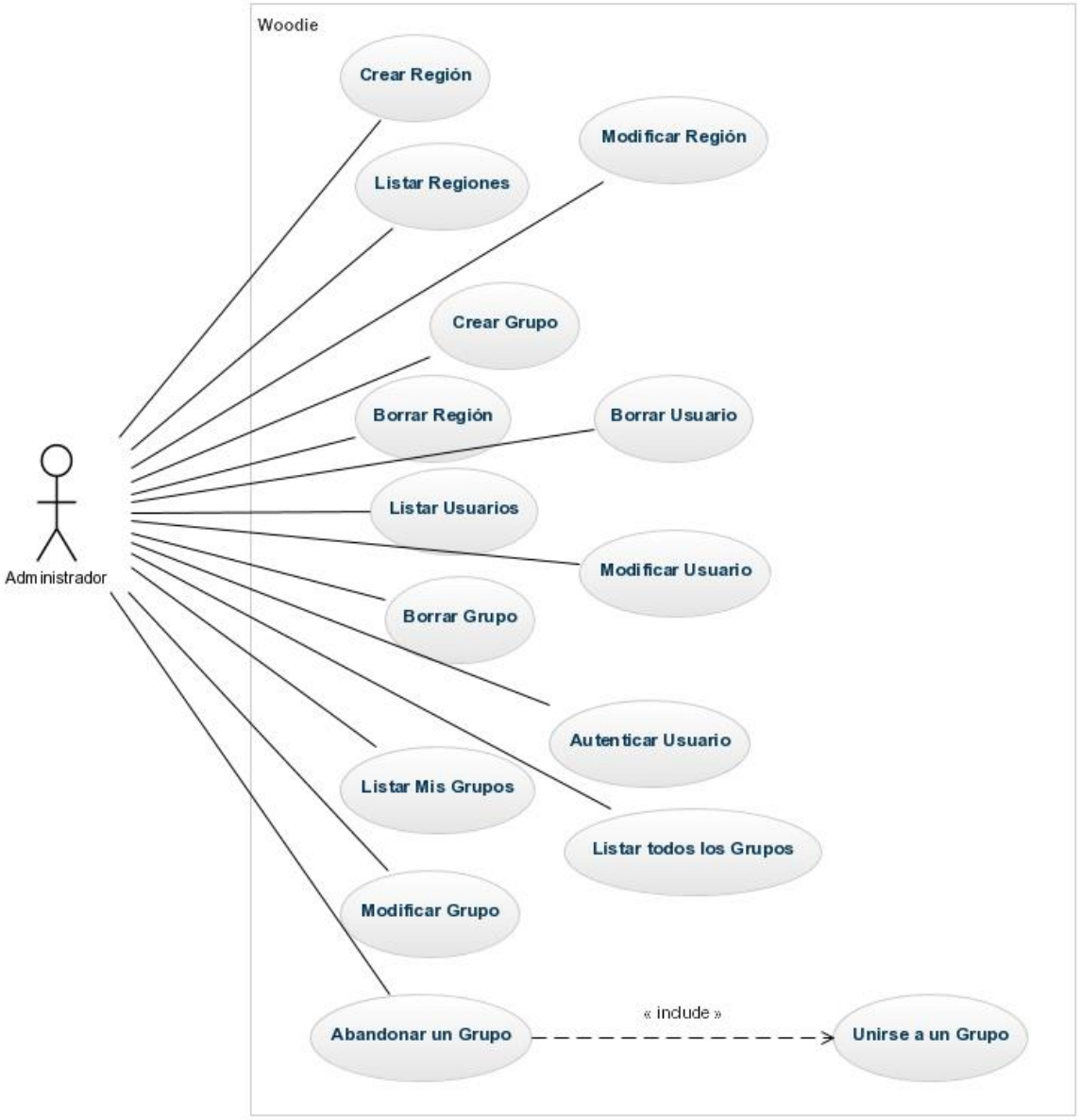

Figura 23. Diagrama de casos de uso del Administrador – Regiones, Grupos y Usuarios

3.7. Requisitos del software

Para facilitar la lectura, los requisitos de software serán recogidos en tablas cuyo formato incluirá los siguientes campos:

- Nombre del requisito: nombre identificativo del requisito.
- Identidicador: es el código único que identifica a cada requisito. Su nomenclatura ha de seguir el formato: SR_WXY_ZZ, donde:
    - XY: es RF si es un requisito funcional y NF si es un requisitono funcional.
    - ZZ : será sustituido por el número de requisito dentro de su categoría.
- Descripción: descripción detallada del requisito correspondiente.
- Origen: indica qué requisito de usuario se evalúa o implementa con dicho requisito software. Cada requisito de usuario tiene que estar cubierto por uno o varios requisitos software.
- Necesidad: indica la importancia del requisito desde el punto de vista del usuario. Los valores pueden ser Esencial y Conveniente.
- Prioridad: indica la prioridad de implementación del requisito. Los valores pueden ser *Baja*, *Media* y *Alta*.
- Estabilidad: indica el grado de variacion que puede experimentar el requisito a lo largo del desarrollo del proyecto. Los valores pueden ser Estable o Inestable.
- Verificabilidad: indica la dificultad que posee la verificacion del cumplimiento del requisito. Los valores pueden ser Baja, Media o Alta.

Los requisitos son los siguientes:

MAPA DE GMAPS	
IDENTIFICADOR: RS_RF_01	
DESCRIPCIÓN: Para hacer más atractiva la aplicación, y mostrar de una manera más clara la localización de los eventos, se integrará GMaps en la aplicación por medio de JQuery.Sólo se podrá marcar una localización por evento.	
ORIGEN: RU_UC_09, RU_UC_10	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

SELECCIONAR FOTO DE USUARIO	
IDENTIFICADOR: RS_RF_02	
DESCRIPCIÓN: Tanto en el registro como en la modificación del perfil, se habilitará un botón para poder seleccionar una imagen como foto de perfil. Si el tamaño de la imagen es superior a 200 Kb se mostrará un mensaje de error al usuario.	
ORIGEN: RU_UC_03, RU_UR_02	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Inestable	**VERIFICABILIDAD:** Alta

AUTOCOMPLETAR NOMBRE DE USUARIO	
IDENTIFICADOR RS_RF_03	
DESCRIPCIÓN: A la hora de crear o modificar un grupo, se pueden añadir nuevos usuarios al grupo. Se habilita mediante AJAX la funcionalidad de autocompletar en una caja de texto. Cuando el usuario introduce un texto, el sistema debe mostrar los usuarios cuyo nombre de usuario concuerdan con el texto introducido.	
ORIGEN: RU_UC_05, RU_UC_06	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

INFORMACIÓN DE PERFIL DISPONIBLE	
IDENTIFICADOR: RS_RF_04	
DESCRIPCIÓN: En la cabecera, siempre se mostrará el nombre del usuario conectado.Al pulsar en el nombre, el sistema mostrará el perfil de dicho usuario, y la opción de poder modificar el perfil.	
ORIGEN: RU_UC_02	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

LISTADOS ORDENADOS Y PAGINADOS	
IDENTIFICADOR: RS_NF_05	
DESCRIPCIÓN: Los listados de regiones, eventos, grupos y usuarios se mostrarán de forma ordenada y paginada,pudiendo el usuario elegir el número de registros que se mostrarán por pantalla	
ORIGEN: RU_UC_11, RU_UC_12, RU_UC_13, RU_UC_14, RU_UC_15, RU_UC_16, RU_UC_20	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE LISTADO DE USUARIOS	
IDENTIFICADOR: RS_NF_06	
DESCRIPCIÓN: En cada listado de usarios,en cada registro se mostrará el nombre de usuario.Además se habilitarán una serie de enlaces: • "Ver": el usuario podrá ver el perfil del usuario. • "Modificar": el usuario podrá modificar el perfil.Sólo está disponible el enlace si el usuario es el mismo o el administrador. • "Borrar usuario": el usuario se elimina de la base de datos.Sólo está disponible el enlace si el usuario es el mismo o el administrador.	
ORIGEN: RU_UC_02, RU_UC_11, RU_UR_05	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE LISTADO DE REGIONES	
IDENTIFICADOR: RS_NF_07	
DESCRIPCIÓN: En cada listado de regiones,en cada registro se mostrará el nombre de la región y el nombre del país. Además, se habilitarán una serie de enlaces: • "Modificar": el administrador podrá modificar los datos de la región. • "Borrar región": el administrador podrá eliminar la región de la base de datos.	
ORIGEN: RU_UC_07, RU_UC_08, RU_UC_14	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE LISTADO DE GRUPOS	
IDENTIFICADOR: RS_NF_08	
DESCRIPCIÓN: En cada listado de grupos, en cada registro se mostrará el nombre del grupo. Además, se habilitarán una serie de enlaces: • "Ver": el usuario puede visualizar información del grupo. • "Modificar": el usuario podrá modificar el grupo.Sólo se muestra el enlace si el usuario es el administrador del grupo o es el administrador. • "Borrar grupo": el grupo es eliminado de la base de datos.Sólo se muestra el enlace si el usuario es administrador del grupo o es el propio administrador. • "Abandonar grupo": el usuario abandona el grupo.Sólo se muestra el enlace si el usuario pertenecía al grupo • "Unirse al grupo": el usuario se une al grupo.Sólo se muestra el enlace si el usuario no pertenece al grupo.	
ORIGEN: RU_UC_06, RU_UC_12, RU_UC_16, RU_UC_18, RU_UR_06	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE LISTADO DE EVENTOS	
IDENTIFICADOR: RS_NF_09	
DESCRIPCIÓN: En cada listado de eventos, en cada registro se mostrará el nombre del evento y el grupo al que pertenece. Además, se habilitarán una serie de enlaces: • "Ver": el usuario puede visualizar información del evento. • "Modificar": el usuario podrá modificar el evento.Sólo se muestra el enlace si el usuario es el administrador del evento o es el administrador. • "Borrar evento": el evento es eliminado de la base de datos.Sólo se muestra el enlace si el usuario es administrador del evento o es el propio administrador. • "Abandonar evento": el usuario abandona el evento.Sólo se muestra el enlace si el usuario estaba apuntado al evento. • "Unirse al evento": el usuario se apunta al evento.Sólo se muestra el enlace si evento pertenece a uno de los grupos en los que está el usuario.	
ORIGEN: RU_UC_10, RU_UC_13, RU_UC_15, RU_UC_17, RU_UC_19, RU_UC_20, RU_UR_04, RU_UR_07	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

ELEMENTOS DEL MENÚ LATERAL	
IDENTIFICADOR: RS_NF_10	
DESCRIPCIÓN: En el menú de opciones lateral se mostrarán las siguientes opciones: • "Grupos": Acceder al menú de grupos. • "Eventos": Acceder al menú de eventos. • "Usuarios":Acceder al listado de usuarios. • "Regiones":Acceder al listado de regiones.Sólo se muestra esta opción en el caso de ser el administrador. • "Novedades": Acceder al listado de nuevos eventos o modificaciones desde la última vez que el usuario accedió al sistema.	
ORIGEN: RU_UR_03	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

BÚSQUEDA EN LISTADOS	
IDENTIFICADOR: RS_RF_11	
DESCRIPCIÓN: En los listados se habilitará un cuadro de texto para hacer búsquedas específicas de algún registro cuyo nombre contenga el texto indicado.Se realizará mediante el uso de AJAX.	
ORIGEN: RU_UC_11, RU_UC_12, RU_UC_13, RU_UC_14, RU_UC_15, RU_UC_16, RU_UC_20	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

GEOLOCALIZACIÓN	
IDENTIFICADOR: RS_RF_12	
DESCRIPCIÓN: En la creación o modificación de eventos se habilitará un cuadro de texto para introducir alguna localización. Al pulsar el botón, el mapa de Google Maps se posicionará en el lugar especificado, si es que existe.	
ORIGEN: RU_UC_09, RU_UC_10	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

MOSTRAR NOVEDADES	
IDENTIFICADOR: RS_NF_13	
DESCRIPCIÓN: Cuando el usuario se autentifica en la aplicación, y accede al menú principal, la interfaz le mostrará un mensaje indicando si se han creado nuevos eventos o ha sido modificado alguno en el que participa. Dicho mensaje viene acompañado con un enlace para mostrar el listado de dichos eventos.	
ORIGEN: RU_UC_17	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

ELIMINAR USUARIO DE GRUPO	
IDENTIFICADOR: RS_RF_14	
DESCRIPCIÓN: En la interfaz de crear o modificar grupo se mostrará un botón “Eliminar” por cada usuario del grupo. Al pulsar el botón, el usuario dejará de pertenecer al grupo.	
ORIGEN: RU_UC_05, RU_UC_06	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

IMPOSIBILIDAD DE CAMBIAR GRUPO DEL EVENTO	
IDENTIFICADOR: RS_NF_15	
DESCRIPCIÓN: En la interfaz de modificación de evento, no se mostrará ninguna opción para modificar el grupo al que pertenece el evento.	
ORIGEN: RU_UR_01	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

EVENTOS ORDENADOS POR FECHA DE SUCESO	
IDENTIFICADOR: RS_NF_16	
DESCRIPCIÓN: En las interfaces que muestran listados de eventos, dichos eventos se mostrarán ordenados por fecha. Los eventos que han sucedido se mostrarán con un color rojo de fondo para diferenciarlos de los demás.Los eventos que suceden en el día actual se mostrarán con color verde.	
ORIGEN: RU_UC_21, RU_UC_22	
NECESIDAD: Esencial	**PRIORIDAD:** Media
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE MENÚ DE GRUPOS	
IDENTIFICADOR: RS_NF_17	
DESCRIPCIÓN: En la interfaz que muestra el menú de grupos se mostrarán los siguientes botones: • "Crear Grupo": al pulsarlo se mostrará la interfaz para crear un grupo. • "Listar Grupos": al pulsarlo de muestra la interfaz con un listado de todos los grupos registrados. • "Mis Grupos": al pulsarlo se muestra la interfaz con un listado de todos los grupos a los que pertenece el usuario.	
ORIGEN: RU_UC_05, RU_UC_12, RU_UC_16	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE MENÚ DE EVENTOS	
IDENTIFICADOR: RS_NF_18	
DESCRIPCIÓN: En la interfaz que muestra el menú de eventos se mostrarán los siguientes botones: • "Crear Eventos": al pulsarlo se mostrará la interfaz para crear un evento. • "Listar Eventos": al pulsarlo se mostrará la interfaz con un listado de todos los eventos registrados. • "Mis Eventos": al pulsarlo se mostrará la interfaz con un listado de todos los eventos a los que el usuario está apuntado. • "Eventos de mis Grupos": al pulsarlo se mostrará la interfaz con un listado de los eventos de los grupos a los que pertenece el usuario.	
ORIGEN: RU_UC_09, RU_UC_13, RU_UC_15, RU_UC_20	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE REGISTRO DE USUARIO	
IDENTIFICADOR: RS_NF_19	
DESCRIPCIÓN: En la interfaz de registro, el usuario debe rellenar una serie de datos para poder estar registrado en el sistema. Los campos obligatorios a rellenar deben ser los de nombre de usuario y password. Además de ellos, se le mostrarán otros campos opcionales como el nombre, apellidos, fecha de nacimiento, ciudad, email, hobbies y mensaje personal. También se le mostrará un selector para elegir el grupo y principal y la posibilidad de asociar una foto al perfil.	
ORIGEN: RU_UC_01,RU_UC_03	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

VISTA DE LOGIN	
IDENTIFICADOR: RS_NF_20	
DESCRIPCIÓN: En la interfaz de login, el usuario estará obligado a introducir los campos de nombre de usuario y contraseña. En caso de que los datos introducidos no existan en la base de datos, se le comunicará al usuario mediante un mensaje.	
ORIGEN: RU_UC_04	
NECESIDAD: Esencial	**PRIORIDAD:** Alta
ESTABILIDAD: Estable	**VERIFICABILIDAD:** Alta

3.8. Matrices de trazabilidad

Para comprobar que todo requisito de usuario está contemplado por al menos un requisito software, utilizaremos una matriz de trazabilidad.

La matriz de trazabilidad tiene en su eje vertical el conjunto de requisitos de usuario, y en su eje horizontal tiene el cojunto de requisitos de software.

La matriz marcará cada requisito de usuario contemplado por un requisito de software concreto. Si una línea de la matriz quedara completamente en blanco, significaría que en la aplicación final no se vería implementado el requisito correspondiente.

La matriz de trazabilidad es la siguiente:

	RS_RF_01	RS_RF_02	RS_RF_03	RS_RF_04	RS_NF_05	RS_NF_06	RS_NF_07	RS_NF_08	RS_NF_09	RS_NF_10	RS_RF_11	RS_RF_12	RS_NF_13	RS_RF_14	RS_NF_15	RS_NF_16	RS_NF_17	RS_NF_18	RS_NF_19	RS_NF_20
RU_UC_01																			X	
RU_UC_02				X		X														
RU_UC_03		X																	X	
RU_UC_04																				X
RU_UC_05			X											X			X			
RU_UC_06			X					X						X						
RU_UC_07							X													
RU_UC_08							X													
RU_UC_09	X											X						X		
RU_UC_10	X								X			X								
RU_UC_11					X	X					X									
RU_UC_12					X			X			X						X			
RU_UC_13					X				X		X							X		
RU_UC_14					X		X				X									
RU_UC_15					X				X		X							X		
RU_UC_16					X			X			X						X			
RU_UC_17									X				X							
RU_UC_18								X												
RU_UC_19									X											
RU_UC_20					X				X		X							X		
RU_UC_21																X				
RU_UC_22																X				

	RS_RF_01	RS_RF_02	RS_RF_03	RS_RF_04	RS_NF_05	RS_NF_06	RS_NF_07	RS_NF_08	RS_NF_09	RS_NF_10	RS_RF_11	RS_RF_12	RS_NF_13	RS_RF_14	RS_NF_15	RS_NF_16	RS_NF_17	RS_NF_18	RS_NF_19	RS_NF_20
RU_UR_01															X					
RU_UR_02		X																		
RU_UR_03										X										
RU_UR_04									X											
RU_UR_05						X														
RU_UR_06								X												
RU_UR_07									X											

Tabla 3. Matriz de trazabilidad de requisitos

Capítulo 4

Diseño

4.1. Introducción

En este capítulo, nos centraremos en el diseño realizado para la aplicación. Para ello, describiremos la arquitectura de la aplicación así como las diferentes capas que interactúan entre ellas. También se analizarán las interfaces que se mostrarán al usuario y la estructura de la base de datos que contiene la información general del sistema.

4.2. Arquitectura del Sistema

La aplicación va a utilizar una arquitectura Cliente-Servidor, que se trata de una arquitectura distribuida en la que el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y éste envía uno o varios mensajes con la respuesta (provee el servicio).

El Cliente va a manejar todas las funciones relacionadas con el formateo y despliegue de datos.Las funciones que lleva a cabo el proceso son los siguientes:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

En cuanto al Servidor, es el encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.Las funciones que lleva a cabo el proceso servidor son los siguientes:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

Por lo tanto, cuando el usuario desea crear grupos o eventos, o bien cuando quiere visualizar un listado de los grupos o eventos ya registrados en el sistema, estará realizando una petición al servidor (para introducir información en la base de datos o para recuperar la información ya existente) .El servidor será el encargado de procesar las peticiones,de introducir los datos en la base de datos o de realizar una consulta y devolver los resultados al cliente.

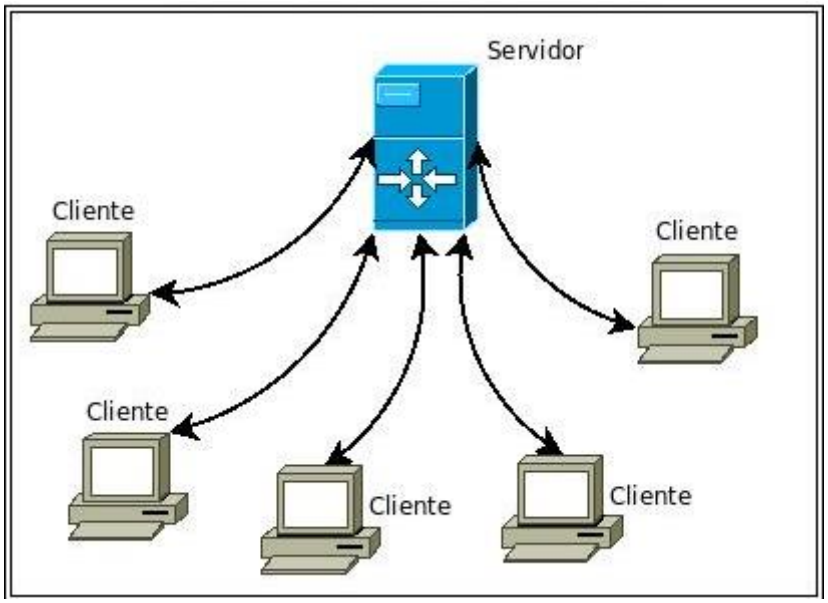

Figura 24. Arquitectura Cliente-Servidor

Se utilizará un diseño Cliente-Servidor en 5 niveles. A continuación veremos cómo estas capas se relacionan entre ellas, y qué responsabilidades poseen.

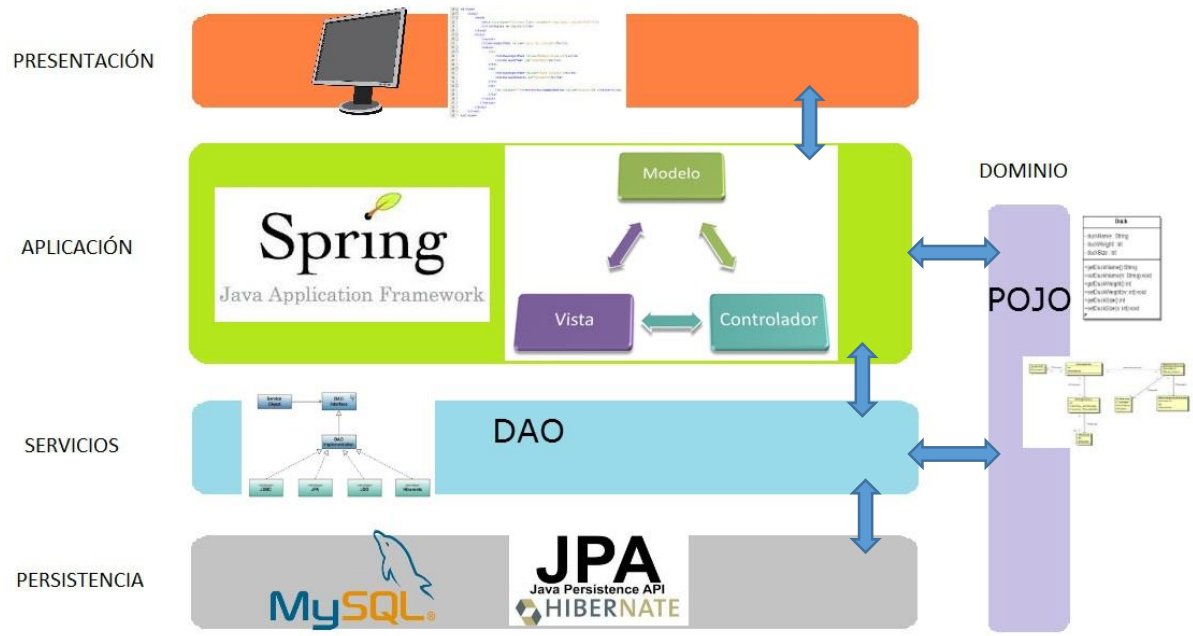

Figura 25. Arquitectura de la aplicación de 5 niveles

Capa presentación

Es la capa con la que interactúa el usuario de la aplicación web, normalmente a través de un navegador. Una de sus tareas, es la de capturar los datos del usuario con los que opera la capa de la aplicación y enviárselos a ésta. Esta capa está formada por el conjunto de las páginas que se pueden ver desde el navegador.

Cuando un usuario se autentica, se crea una sesión automáticamente con los datos del usuario, esto le permite al usuario acceder a diferentes partes de la aplicación sin tener que preocuparse de autenticarse otra vez.

Para el diseño de las vistas, se han utilizado páginas JSP. Se han utilizado hojas de estilo CSS para formatear el contenido y así aprovechar las ventajas que este sistema nos ofrece, como tener limpio el código HTML y tener en varios ficheros de configuración el aspecto que debe tener la aplicación, permitiéndonos realizar cambios en toda la aplicación únicamente modificando una línea de código en estos ficheros CSS, esto promueve la usabilidad y la accesibilidad.

Se ha utilizado también el framework Tiles para el diseño de elementos comunes a las páginas, como son la cabecera,el menú lateral,etc.

Capa aplicación

En esta capa, se encuentra la aplicación en sí. Ésta se encuentra en el servidor a la que acceden los usuarios a través de la red .las funciones de la capa Aplicación consisten en recoger los datos enviados desde la capa de presentación, procesar la información, implementar la lógica de la aplicación, acceder a los datos y generar las respuestas para el cliente.

También se encarga de interaccionar con la capa de servicios para acceder a la capa de persistencia.

Es en esta capa donde Spring y su patrón MVC entra en juego, además de utilizar la funcionalidad de la inyección de dependencias.

Capa Servicios

Esta capa actúa como un puente entre las capas de aplicación y persistencia usando la capa de dominio como contenedor de información, también llamados DTO (Data Transfer Protocol).

Estos servicios de negocio, usan el patrón DAO (Data Access Object) para acceder a la información de la capa de persistencia mediante una interface, de esta forma conseguimos que nuestra lógica de negocio (aplicación) no sepa nada de Hibernate, y siempre que quiera acceder a los datos lo hará usando la interface DAO, con esto conseguimos reducir el acoplamiento y así podemos intercambiar la implementación fácilmente si algún día nos cansamos de Hibernate/JPA sin que eso repercuta en el código de la aplicación. En los anexos, se explicará con más detalle cómo funciona el patrón DAO.

Capa Dominio

Esta capa, se comunica directamente con las capas de aplicación, servicios y persistencia. Hibernate usará estos objetos de dominio, que serán persistidos en en nuestra base de datos relacional. Para ello Hibernate ha de conocer como relacionar el objeto con la base de datos, para eso dispone de una serie de anotaciones que utilizará para definir el objeto a persistir.

Capa Persistencia

La función básica de esta capa, es la persistencia de un modelo de dominio basado en objetos dentro de una base de datos relacional (MySQL) mediante un ORM, que en este caso es Hibernate/JPA.

Hibernate, será el encargado de acceder, modificar o eliminar registros en la base de datos de Mysql. La única configuración que requiere MySQL, es su instalación y creación de una base de datos. La función de creación de tablas y relaciones, está delegada a Hibernate.

4.3. Diseño de la base de datos

En la aplicación será necesario persistir la información en un fuente de datos, para poder acceder a ella en cualquier momento. Para ello se utilizará MySql como gestor de base de datos.También se utilizará Hibernate para facilitarnos el mapeo de atributos entre la base de datos y el modelo de clases de la aplicación.

Cada aplicación necesita clases que representen los conceptos principales de la aplicación que será desarrollada. Estas clases contienen y manejan varias relaciones entre ellas, juntas forman el diagrama de clases de la aplicación. El diagrama de clases ha sido desarrollado bajo UML – Unified Modeling Language.

El diagrama es el siguiente:

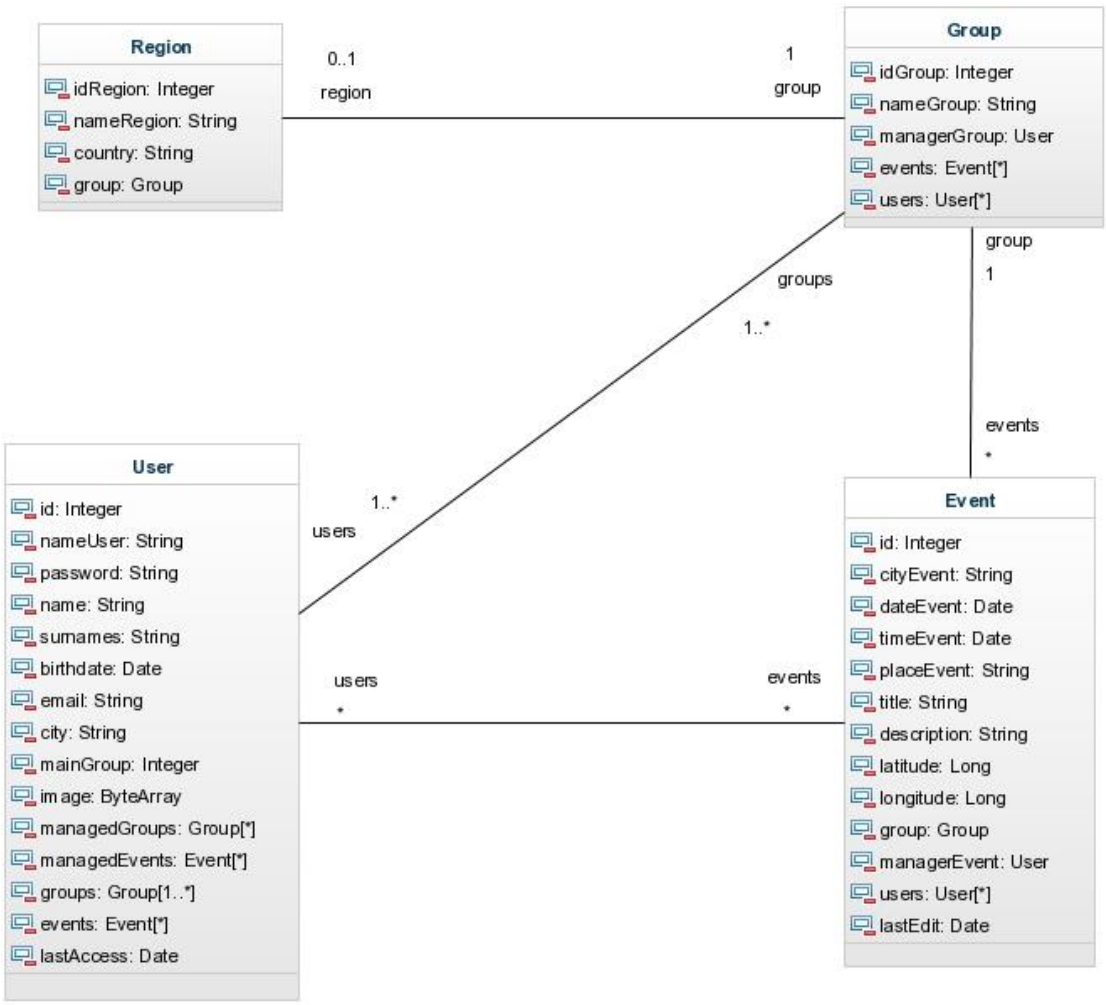

Figura 26. Diagrama de clases

Vamos a explicar en qué consisten cada una de las clases del diagrama y sus atributos.

Region

Esta tabla almacena la información sobre la distintas regiones.Un usuario al registrarse sólo pertenecerá a un grupo inicial, que serán los grupos asociados a las regiones que existen.Sólo el administrador tendrá acceso a estos datos.

- idRegión: clave que identifica la región.
- nameRegion: nombre de la región.
- country: nombre del país al que pertenece la región.
- group: grupo asociado a la región.

Group

Esta tabla almacena la información sobre los grupos registrados. Aparte de los grupos iniciales (asociados de las regiones), los usuarios podrán crear nuevos grupos en los que se crearán eventos y se añadirán usuarios a ellos.

- idGroup: clave que identifica al grupo.
- nameGroup: nombre del grupo.
- managerGroup: el usuario que creó eñ grupo, y el único que podrá editarlo.
- events: lista de eventos que han sido creados en el grupo.
- users: lista de usuarios que pertenecen al grupo.El administrador al crear el grupo está incluido en la lista por defecto.

User

Esta tabla almacena la información sobre los usuarios.Cada usuario registrado en el sistema es guardado en esta tabla.

- id: clave del usuario.
- nameUser: nombre de usuario que lo identificará de manera única en el sistema.
- password: contraseña que junto con el nombre de usuario ayuda a identificar a un usuario.
- name: nombre del usuario.
- surnames: apellidos del usuario.
- birthdate: fecha de nacimiento del usuario.
- email: correo electrónico del usuario.
- city: ciudad en la que vive el usuario
- mainGroup: grupo inicial a la que va a pertenecer el usuario.Lo tendrá que seleccionar entre una lista de los disponibles.
- image: imagen que aparece en su perfil de usuario.
- managedGroups: lista de grupos administrados por el usuario.
- managedEvents: lista de eventos administrados por el usuario.

- groups: lista de grupos a los que pertenece el usuario.
- events: lista de eventos a los que está apuntado el usuario.
- lastAccess: fecha y hora de la última vez que el usuario accedió al sistema.

Event

Esta tabla almacena la información sobre los eventos. Un evento pertenece a un solo grupo y los usuarios se apuntarán a los eventos que consideren interesantes.

- id: clave del evento.
- cityEvent: ciudad en la que sucederá el evento.
- dateEvent: fecha en la que sucederá el evento.
- timeEvent: hora en la que sucederá el evento.
- placeEvent: descripción del lugar donde sucederá el evento.
- title: título del evento.
- description: información detallada del evento.
- latitude: latitud marcada en el mapa de Google Maps
- longitude: longitud marcada en el mapa de Google Maps.
- group: grupo a que pertenece el evento.
- managerEvent: administrador del evento. Será el único que podrá editar el evento.
- users: lista de usuarios apuntados al evento.
- lastEdit: fecha de creación o de última modificación del evento.

4.4. Diseño de la interfaz

En este apartado analizaremos el diseño de la interfaz de la aplicación.

El principal objetivo es conseguir una interfaz intuitiva, atractiva y fácil de usar, que muestre la información que necesita el usuario de la forma más sencilla posible y que éste consiga sus objetivos en el menor número de pasos posibles, puesto que uno de los objetivos que pretende la aplicación para diferenciarse de otras redes sociales más complejas es la sencillez y rapidez en la realización de las tareas.

A continuación se explican las decisiones de diseño de la interfaz principal:

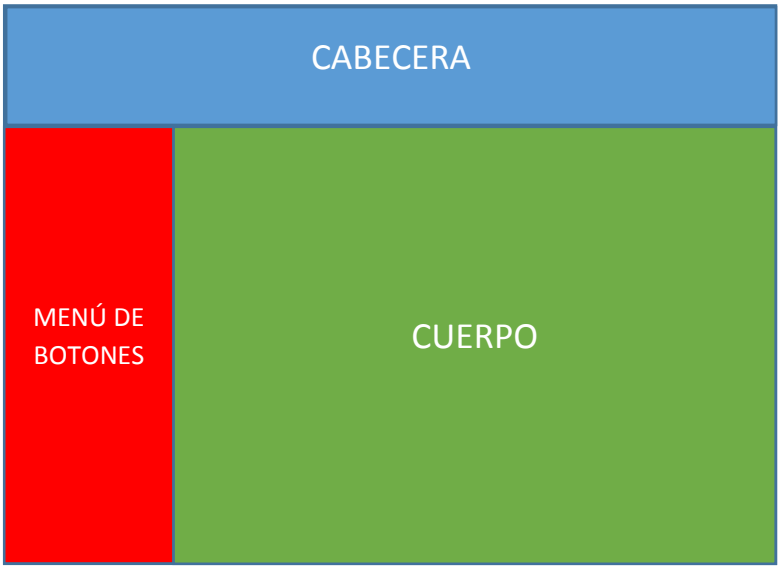

Figura 27. Plantilla de interfaz

Se ha dividido la interfaz en 3 secciones: cabecera, menú de botones y cuerpo.

En la sección de cabecera se muestra el logo de la aplicación, y también se mostrará el nombre del usuario conectado, además de mostrar el botón para desconectarse del sistema. Al pulsar en el nombre de usuario, se mostrará la interfaz de visualizar perfil.

En la sección de menú de botones, se mostrarán los botones que dan acceso a las principales secciones de la aplicación, como son la gestión de usuarios, la gestión de eventos y la gestión de grupos, además de la gestión de regiones (si es el administrador).

En la sección del cuerpo se irán mostrando los distintos formularios o listados que la aplicación pondrá a servicio del usuario.

Capítulo 5

Implementación

Debido a que el desarrollo de la aplicación gira en torno al framework de Spring, va a ser este framework quien nos va ayudar en la implementación de la misma, aportándonos todas sus funcionalidades y beneficios como la inyección de dependencias.

Además, Spring nos permite mantener el código desacoplado, limpio y reutilizable, ya que la filosofía de Spring nos guía a programar orientado a interfaces.

El utilizar frameworks como Spring e Hibernate y hacer uso de las anotaciones nos va a permitir reducir el tiempo de desarrollo con Java.

Por lo tanto, una de las tareas más importantes consistirá en la configuración de Spring y el permitir el uso de anotaciones.

A partir de Spring 2.5 la manera más interesante de enlazar nuestros beans es a través de las anotaciones.

El uso de anotaciones nos permite eliminar el tener la definición de nuestros beans en un archivo XML, y esto es útil cuando tenemos declarados muchos beans y nuestro archivo se vuelve engorroso (aunque siempre se pueden combinar ambas formas de declaración).

Para activar la configuración de Spring para que detecte las anotaciones generales (como son el caso de @Autowired o @Required) se usa el elemento <context:annotation-config/>. Para indicar en qué paquetes se encuentran las clases que hemos anotado usamos el elemento "component-scan".Mediante su elemento "base-package",indicamos el paquete raíz en el que se encuentran nuestros beans anotados.

La declaración <mvc:annotation-driven> habilita el soporte de anotaciones para el módulo Spring MVC .También habilitan soporte para las anotaciones utilizadas en la conversión, formateo y validación.

Para activar la configuración de repositorios hay que indicar el paquete a partir del cuál Spring debe buscar clases que extiendan de Repository.

Por lo tanto, nuestro archivo de configuración va a incluir las siguientes líneas de código:

```
<context:annotation-config />

<!-- Scans the classpath of this application for @Components to
deploy as beans -->
<context:component-scan base-package="src" />

<jpa:repositories base-package="repositories" />

 <!-- Enables the Spring MVC @Controller programming model -->
<mvc:annotation-driven />
```

Figura 28. Código del archivo de configuración

Spring proporciona una serie de anotaciones con las cuales podemos indicar exactamente cómo queremos que se manejen los componentes. Para esto existen tres anotaciones básicas:

- @Repository : indica que las clases marcadas con esta anotación están relacionada de alguna forma con una capa de persistencia de datos.
- @Service : indica que las clases marcadas con esta anotación están en una capa de servicios o de lógica de negocios.
- @Controller : indica que las clases marcadas con esta anotación son el controlador de una aplicación web.

Las tres anotaciones anteriores extienden de "@Component", la cual indica que la clase es un componente Spring, y por lo tanto son candidatas para ser auto-detectadas cuando usamos una configuración basada en anotaciones.

Por lo tanto, en la implementación de la aplicación vamos a usar anotaciones y también vamos a apoyarnos en los patrones MVC y DAO.

Spring nos ayuda en este sentido, porque uno de sus módulos más importantes es Spring MVC, el cual nos provee un exhaustivo soporte para el patrón MVC.

En las clases que forman parte del modelo, podemos agregar a nuestro código anotaciones para que puedan hacer uso de JPA, y con ello llevar a cabo una implementación automática de las tablas de nuestra base de datos.

El siguiente es un ejemplo de una de las clases del modelo de la aplicación usando anotaciones:

```java
@Entity
@Table(name="users")
public class User implements Serializable{

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;

	@NotEmpty
	@Column(nullable = false)
	private String nameUser;

	@NotEmpty
	@Column(nullable = false)
	private String password;

	private String name;

	private String surnames;

	@DateTimeFormat(pattern="dd/MM/yyyy")
	private Date birthDate;

	@Email
	private String email;

	private String city;

	private Integer mainGroup;

	@Lob
	private byte[] image;

	private String personalMessage;

	private String hobbies;

	@Column(nullable = false)
	private Date lastAccess;

	@OneToMany(cascade=CascadeType.REMOVE, fetch=FetchType.EAGER,
	mappedBy="managerGroup")
	private Set<Group> managedGroups = new HashSet<Group>();

	@OneToMany(cascade=CascadeType.REMOVE, fetch=FetchType.EAGER,
	mappedBy="managerEvent")
	private Set<Event> managedEvents = new HashSet<Event>();

	@ManyToMany(fetch = FetchType.EAGER)
	@JoinTable(name="users_events",
	joinColumns=@JoinColumn(name="user_id"),
	inverseJoinColumns=@JoinColumn(name="event_id"))
	private Set<Event> events = new HashSet<Event>();

	@ManyToMany(fetch = FetchType.EAGER ,mappedBy="users")
	private Set<Group> groups = new HashSet<Group>();
```

Figura 29. Código de la clase User del modelo

Vamos a explicar para qué sirven las anotaciones incluidas en la clase:

- @Entity para identificar el bean como una entidad a guardar en la base de datos
- @Table indicamos el nombre que tendrá la tabla en la base de datos.
- @Id para identificar el campo de clave primaria.
- @GeneratedValue para permitir a la base de datos asignar automáticamente un valor a la clave primaria cuando un registro es insertado.
- @NotEmpty para indicar que la cadena no puede ser vacía.
- @Column para indicar propiedades de la columna.En este caso indicamos que el valor de la columna no puede ser nulo.
- @DateTimeFormat para el formateo de fechas.
- @Email para comprobar que el email introducido cumple el formato.
- @Lob para el uso de imágenes.
- @OneToMany y @ManyToMany indican el tipo de relación que va a tener la entidad con otras entidades.
- @JoinTable y @JoinColumn para indicar el nombre que va a tener la tabla y las columnas de las tablas intermedias.

En la parte del Controlador, también el uso de las anotaciones nos ayuda con el desarrollo de las clases. El siguiente es un ejemplo de un extraxto de código de uno de los controladores de la aplicación:

```
@Controller
public class EditGroupController {

	@Autowired
	private IGroupService groupService;

	@Autowired
	private IUserService userService;

	private List<Integer> usersList =new ArrayList<Integer>();

	private List<User> usList;

	private User manager = new User();

	private String control;


	@RequestMapping(value = "/editGroup",method = RequestMethod.GET)
	public String edit(@RequestParam(value = "idGroup")String id,
			@RequestParam(value = "controller")String controller,
			ModelMap model,HttpServletRequest request){


		Integer idGroup = Integer.parseInt(id);
		control = controller;
		Group group = groupService.getGroup(idGroup);
		manager = group.getManagerGroup();
		List<User> list = new ArrayList<User>();
		for(User us :group.getUsers()){
			list.add(us);
		}
		usList = list;

		model.put("list", list);
		model.put("group",group);

	    return "editGroup";
	}
```

Figura 30.Código de un controlador

Las anotaciones utilizadas son las siguientes:

- @Controller: esta anotación la emplea Spring para escanear las clases y así detectar cuales de estas clases forman los Controllers de nuestra aplicación. Por lo tanto, el contenedor de Spring detectará automáticamente esta clase y sabrá que la clase es un controlador de Spring MVC capaz de manejar solicitudes web.
- @Autowired nos sirve para llevar a cabo la inyección de dependencias. Nos sirve por lo tanto para inyectar un Bean usando la autodetección de Spring. En este caso inyectaremos un bean de tipo "IUserService" en el atributo userService.
- @RequestMapping la emplea Spring para conocer a qué Controller o método de un Controller tiene que direccionar cada llamada del cliente. En este caso, le estamos informando de que todas las llamadas a "editGroup.htm" van a ser manejadas por el método edit de nuestro Controller.
- @RequestParam la indicamos para recibir parámetros.Los métodos del controlador pueden recibir parámetros de una petición del navegador, bien sea por método POST o por método GET. En este caso, indicamos que vamos a recibir 2 parámetros, uno llamado "idGroup" y otro llamado "controller".

También podemos distinguir diferentes tipos de llamadas a una misma dirección, y tratarlas según nos convenga.

De este modo, si la llamada a la dirección se realiza por GET o POST la trataremos de diferente forma. En este caso estamos recibiendo una petición HTTP de tipo GET.Otro método de esta clase se encarga de recibir las peticiones HTTP de tipo POST.

También nos podemos fijar que el método devuelve un String. Este String hace referencia al nombre del archivo JSP que será llamado cuando acabe la ejecución del método.

En la vistas, se han utilizado las etiquetas que proporciona JSTL, que es una colección de funciones de uso común cuando se desarrollan páginas dinámicas. El uso de estas funcionalidades permiten que las páginas sean más fáciles de leer y mantener. El siguiente código es un ejemplo de vista JSP utilizando etiquetas JSTL:

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<div class="region_form">
      <h3>Modificar Región</h3>
      <br/>
      <c:if test="${not empty error}">
            <label>Error: ${error}</label>
      </c:if>
      <form:form id="regionForm" modelAttribute="region" method="POST">
            <div class="form_row">
                  <form:label path ="country">Nombre Pais (*):</form:label>
                  <form:input path="country" class="input_form"/>
                  <font color="red"><form:errors path="country"/></font>
            </div>
            <div class="form_row">
                  <form:label path ="nameRegion">Nombre region (*):</form:label>
                  <form:input path="nameRegion" class="input_form"/>
                  <font color="red"><form:errors path="nameRegion"/></font>
            </div>
            <label>(*) Campos obligatorios</label>
            <input class="myButtonRed" type="submit" value="Editar Region"/>
      </form:form>
</div>
```

Figura 31. Código de una vista JSP

En este caso se utilizan las etiquetas c y form. La etiqueta c se utiliza para realizar iteración sobre datos y operaciones condicionales. La etiqueta form se utiliza para realizar operaciones específicas del formulario como indicar qué objeto del modelo de datos vamos a enlazar con el formulario, indicar qué atributos del objeto se enlazan con cada elemento del formulario o mostrar errores.

Como se dijo anteriormente, en la aplicación se usa el patrón de diseño DAO. Una buena práctica es crear un DAO genérico que será usado en todos los DAO's específicos debido a que hay operaciones comunes en todos los DAO´s. El DAO genérico implementará métodos como buscar, modificar, guardar o borrar.

El código del DAO genérico es el siguiente:

```
@Repository
public abstract class GenericDAOImpl<Entity, K extends Serializable>
implements IGenericDAO<Entity, K> {

	private Class<Entity> type;


	@PersistenceContext
	private EntityManager entityManager;

	public GenericDAOImpl() {
	  this.type = (Class<Entity>) ((ParameterizedType)
	  getClass().getGenericSuperclass()).getActualTypeArguments()[0]
	  ;
	}


	public Entity find(K id) {
		return (Entity) this.entityManager.find(type, id);
	}


	public void update(Entity entity)  {
		this.entityManager.merge(entity);
	}


	public void save(Entity entity)  {
		this.entityManager.persist(entity);
	}


	public void delete(Entity entity)  {
		this.entityManager.remove(entity);
	}
}
```

Figura 32. Código del DAO genérico

Se puede observar que la clase tiene la anotación @Repository, lo cual indica que es una clase relacionada con la capa de persistencia (una clase DAO).
También se usa la anotación @PersistenceContext, mediante la cual podemos obtener una referencia al EntityManager. El contenedor de Spring nos proporciona el contexto de persistencia mediante inyección por lo que no tendremos que preocuparnos de su creación y destrucción.

Capítulo 6

Pruebas

Las pruebas pueden determinar el éxito o fracaso de un producto final, ya que deben proporcionarnos garantías sobre el funcionamiento correcto de la aplicación.

Para asegurar la estabilidad y robustez, se han realizado diversas pruebas,tanto funcionales como para comprobar el tiempo de respuesta de la aplicación.

6.1. Pruebas de funcionalidad

Estas pruebas se han realizado para comprobar que la aplicación responde a los requisitos establecidos en la fase de análisis.

Por cada operación se ha intentado tratar con todos los casos posibles, a fin de garantizar que el resultado no permita situaciones que escapen al control del usuario final.

Del mismo modo,cuando se han encontrado errores, estos han sido corregidos y las pruebas se han vuelto a realizar hasta encontrar los resultados esperados.

Para facilitar la lectura de las pruebas, éstas son mostradas en tablas.El formato de las tablas es el siguiente:

- Nombre: nombre descriptivo de la prueba.
- Identificador: identificador de la prueba. Seguirá el formato PF_XX, donde XX corresponde con el número de la prueba.
- Descripción: descripción del objetivo de la prueba.
- Acciones: pasos a seguir para la realización de la prueba.
- Resultado: especifica si el resultado ha sido el esperado o se han encontrado errores.Los valores pueden ser Correcto e Incorrecto.

CONEXIÓN CON SERVIDOR
IDENTIFICADOR: PF-01
DESCRIPCIÓN: Comprobar que el cliente se conecta con el servidor y la aplicación inicia adecuadamente.
ACCIONES: 1. Iniciar la aplicación cliente. 2. Introducir dirección web y puerto del servidor y continuar. 3. Comprobar que la aplicación ha iniciado de forma correcta y se muestra la pantalla de inicio.
RESULTADO: Correcto

MENÚS PRINCIPALES
IDENTIFICADOR: PF-02
DESCRIPCIÓN: Comprobar que se muestran correctamente los menús de grupos y eventos.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón “Grupos” del menú lateral y comprobar que se muestra el menú de grupos correctamente. 3. Pulsar el botón “Eventos” del menu lateral y comprobar que se muestra el menú de eventos correctamente.
RESULTADO: Correcto

NOTIFICACIÓN DE NOVEDADES
IDENTIFICADOR: PF-03
DESCRIPCIÓN: Comprobar que se muestra un mensaje con las novedades de eventos al autenticarse en la aplicación.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Crear/modificar un evento. 3. Desconectarse de la aplicación. 4. Iniciar sesión con otro usuario que pertenezca al grupo del evento creado o esté apuntado al evento modificado. 5. Comprobar que se muestra un mensaje en pantalla con las novedades en cuanto a eventos. 6. Pulsar el enlace que acompaña al mensaje para comprobar el listado de los eventos creados o modificados.
RESULTADO: Correcto

CREAR EVENTO USANDO EL MAPA DE GMAPS
IDENTIFICADOR: PF-04
DESCRIPCIÓN: Comprobar que el mapa de Google Maps funciona correctamente al crear un evento.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de Eventos y posteriormente al de Crear Evento. 3. Introducir datos para crear el evento y pulsar una localización en el mapa de Google Maps y comprobar que se muestra un marcador.Pulsar en otro lugar del mapa para comprobar que no se añade ningún nuevo marcador.
RESULTADO: Correcto

ASOCIAR FOTO EN EL REGISTRO DE USUARIO
IDENTIFICADOR: PF-05
DESCRIPCIÓN: Comprobar que se puede asociar una foto en el registro de usuario y que cumple las restricciones de tamaño.
ACCIONES: 1. Iniciar la aplicación y pulsar el botón de registro. 2. Rellenar los datos de registro de usuario y asociar una foto que ocupe más de 200 Kb y pulsar el botón de registrar. 3. Comprobar que se muestra un mensaje de error indicando que el tamaño no cumple las restricciones. Asociar una imagen que ocupe menos de 200 Kb y pulsar el botón de registrar usuario. 4. Comprobar que el registro se ha realizado correctamente.
RESULTADO: Correcto

AGREGAR/ELIMINAR USUARIOS A GRUPOS
IDENTIFICADOR: PF-06
DESCRIPCIÓN: Comprobar que a la hora de crear o modificar grupos funciona la funcionalidad de autocompletar y se añaden y eliminan los usuarios correctamente.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de Grupos y luego al de Crear Grupo. 3. Rellenar los datos del grupo.Introducir un texto en la caja de texto de buscar Usuario y comprobar que aparece la funcionalidad de autocompletar. 4. Pulsar el botón de agregar usuario y comprobar que el usuario se ha añadido correctamente. 5. Pulsar el botón de eliminar con algún usuario del grupo y comprobar que se ha eliminado. 6. Pulsar el botón de Crear Grupo y comprobar que el grupo se ha creado con los usuarios añadidos.
RESULTADO: Correcto

PERFIL DE USUARIO DISPONIBLE EN CABECERA
IDENTIFICADOR: PF-07
DESCRIPCIÓN: Comprobar que al pulsar el nombre del usuario conectado situado en la cabecera se muestra el perfil.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Comprobar que se muestra el nombre del usuario en la cabecera. 3. Pulsar el nombre del usuario y comprobar que se muestra el perfil de forma correcta y, además, se muestra un botón para modificar el perfil.
RESULTADO: Correcto

ORDENACIÓN Y PAGINACIÓN EN LISTADOS
IDENTIFICADOR: PF-08
DESCRIPCIÓN: Comprobar que se muestran todos los listados de forma ordenada y paginada, y que se pueden realizar búsquedas.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de Regiones y comprobar que el listado se muestra de forma ordenada y paginada. 3. Introducir un texto en la caja de texto para realizar búsquedas y pulsar el botón.Comprobar que se muestran los registros con el patrón que se introdujo. 4. Realizar la misma acción con grupos, eventos y usuarios.
RESULTADO: Correcto

VISUALIZACIÓN DE ENLACES EN LISTADOS DE GRUPOS
IDENTIFICADOR: PF-09
DESCRIPCIÓN: Comprobar que se muestran los enlaces adecuados en los listados de grupos, es decir,que no aparezcan enlaces como los de modificar o borrar cuando el usuario conectado no es administrador.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de grupos y acceder al listado. Comprobar que en los enlaces de cada registro sólo aparecen los enlaces de modificar y borrar de los grupos administrados. De igual manera, deben aparecer los enlaces de abandoner grupo en los grupos a los que pertenecemos y el de unirse al grupo en los grupos a los que no pertenecemos.
RESULTADO: Correcto

VISUALIZACIÓN DE ENLACES EN LISTADOS DE EVENTOS
IDENTIFICADOR: PF-10
DESCRIPCIÓN: Comprobar que se muestran los enlaces adecuados en los listados de eventos, es decir, que no aparezcan enlaces como los de modificar o borrar cuando el usuario conectado no es administrador.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de eventos y acceder al listado. Comprobar que en los enlaces de cada registro sólo aparecen los enlaces de modificar y borrar de los eventos administrados. De igual manera, deben aparecer los enlaces de abandoner evento en los eventos a los que pertenecemos y el de unirse al evento en los eventos a los que no pertenecemos, además de ser eventos pertenecientes a un grupo nuestro.
RESULTADO: Correcto

VISUALIZACIÓN DE ENLACES EN LISTADOS DE USUARIOS
IDENTIFICADOR: PF-11
DESCRIPCIÓN: Comprobar que se muestran los enlaces adecuados en los listados de usuarios, es decir, que no aparezcan enlaces como los de modificar o borrar cuando el usuario conectado no es el propio usuario del registro.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de usuarios y acceder al listado. Comprobar que en los enlaces de cada registro no aparecen los enlaces de modificar o borrar perfil, y que sólo aparece en el registro del propio usuario.
RESULTADO: Correcto

COMPROBAR GEOLOCALIZACIÓN
IDENTIFICADOR: PF-12
DESCRIPCIÓN: Comprobar que la funcionalidad de geolocalización funciona en los mapas de GMaps.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de eventos y acceder a la creación de eventos.Introducir un lugar (debe existir) en la caja de texto de la geolocalización y comprobar que el mapa de GMaps se posiciona en dicho lugar.
RESULTADO: Correcto

LISTADO DE EVENTOS ORDENADOS
IDENTIFICADOR: PF-13
DESCRIPCIÓN: Comprobar que el listado de eventos se encuentra ordenado por fecha y que los eventos ya sucedidos y los del día actual se muestran con el color adecuado.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de eventos y acceder al listado de eventos.Comprobar que se encuentran ordenados por fecha de suceso y se muestran con los colores correspondientes.
RESULTADO: Correcto

IMPOSIBILIDAD DE CAMBIAR GRUPO DE EVENTO
IDENTIFICADOR: PF-14
DESCRIPCIÓN: Comprobar que no se permite cambiar el grupo de un evento ya creado.
ACCIONES: 1. Iniciar la aplicación y autenticarse como usuario. 2. Pulsar el botón de eventos y acceder a la creación de eventos.Introducir los datos para crear un evento y pulsar el botón de crear evento. 3. Acceder al listado de eventos y pulsar en modificar evento del evento creado en el paso 2. 4. Comprobar que no se permite modificar el grupo
RESULTADO: Correcto

LOGIN
IDENTIFICADOR: PF-15
DESCRIPCIÓN: Comprobar que no se permite acceder a la aplicación a los usuarios no registrados.
ACCIONES: 1. Iniciar la aplicación y acceder a la pantalla de login. 2. Rellenar el nombre de usuario con un usuario válido y el password incorrecto. 3. Comprobar que al pulsar el botón Login se muestra un mensaje indicando que los datos no son correctos.
RESULTADO: Correcto

6.2. Pruebas de rendimiento

Las pruebas de rendimiento se realizan para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo.
Estas pruebas de rendimiento se van a mostrar en tablas, las cuales seguirán el siguiente formato.

- Nombre: nombre descriptivo de la prueba.
- Identificador: identificador de la prueba. Seguirá el formato PR_XX, donde XX corresponde con el número de la prueba.
- Descripción: descripción del objetivo de la prueba.
- Resultado: resume los resultados obtenidos al realizar la prueba.

PRUEBA CREAR EVENTO
IDENTIFICADOR: PF-01
DESCRIPCIÓN: Prueba que consiste en insertar simultáneamente en base de datos 5, 10, 15 y 20 eventos con una conexión a Internet de 20 Mbps.
RESULTADO: Los resultados obtenidos son los siguientes: • 5 eventos: 0,5 segundos. • 10 eventos: 0,8 segundos. • 15 eventos: 1,3 segundos. • 20 eventos: 1,7 segundos.

PRUEBA CREAR GRUPO
IDENTIFICADOR: PF-02
DESCRIPCIÓN: Prueba que consiste en insertar simultáneamente en base de datos 5, 10, 15 y 20 grupos con una conexión a Internet de 20 Mbps.
RESULTADO: Los resultados obtenidos son los siguientes: • 5 grupos: 0,6 segundos. • 10 grupos: 0,9 segundos. • 15 grupos: 1,5 segundos. • 20 grupos: 1,9 segundos.

PRUEBA VISUALIZAR LISTADOS
IDENTIFICADOR: PF-03
DESCRIPCIÓN: Prueba que consiste en visualizar 5,10,15 y 20 listados simultáneamente con una conexión a Internet de 20 Mbps.
RESULTADO: Los resultados obtenidos son los siguientes: • 5 listados: 0,5 segundos. • 10 listados: 0,9 segundos. • 15 listados: 1,6 segundos. • 20 listados: 2,1 segundos.

6.3 Matriz de trazabilidad de funcionalidad

La matriz de trazabilidad de funcionalidad tiene el objetivo de comprobar que los requisitos software recogidos en la sección de análisis hayan sido implementados en el sistema y funcionen correctamente.

Para ello, se marcan las intersecciones entre requisitos y pruebas en aquellas pruebas en las que se puede demostrar que el requisito software se ha cumplido.

La matriz de trazabilidad tiene en su eje vertical el conjunto de requisitos de software, y en su eje horizontal tiene el cojunto de pruebas funcionales.

Para poder asegurar que las pruebas han validado los requisitos del sistema, todas las filas de la matriz deben tener al menos una marca.

	PF_01	PF_02	PF_03	PF_04	PF_05	PF_06	PF_07	PF_08	PF_09	PF_10	PF_11	PF_12	PF_13	PF_14	PF_15
RS_RF_01				X											
RS_RF_02					X										
RS_RF_03						X									
RS_RF_04							X								
RS_NF_05								X							
RS_NF_06								X			X				
RS_NF_07								X							
RS_NF_08								X	X						
RS_NF_09								X		X					
RS_NF_10		X													
RS_RF_11								X							
RS_RF_12												X			
RS_NF_13			X												
RS_RF_14						X									
RS_NF_15														X	
RS_NF_16													X		
RS_NF_17		X													
RS_NF_18		X													
RS_NF_19					X										
RS_NF_20	X														X

Tabla 4. Matriz de trazabilidad de funcionalidad

Capítulo 7

Conclusiones y líneas futuras

En esta sección, se analizarán la consecución de los objetivos para el proyecto y las conclusiones e ideas obtenidas a lo largo de su desarrollo.
También se explicarán las metas personales que se han conseguido y la experiencia obtenida en el desarrollo del proyecto.
Por último, se explicarán ideas y mejoras a realizar en el proyecto en un futuro próximo.

7.1. Conclusiones del proyecto

La realización del proyecto, que consistía en desarrollar una aplicación que se centrara en la gestión de eventos y grupos, de una forma más sencilla y simple a como lo hacen otras redes sociales, se ha conseguido satisfactoriamente.

Se pretendía diseñar una aplicación web que siguiera la tendencia web 2.0. Para ello se han utilizado tecnologías de desarrollo web actuales que ayudaran al desarrollador en su labor, centrando el desarrollo en el framework Spring, uno de los más extendidos en el mercado actual y apoyándonos en otros frameworks muy utilizados como son Hibernate y Tiles.

Se ha podido comprobar como el utilizar estos frameworks J2EE han ayudado evidentemente al desarrollo de la aplicación, y el uso de anotaciones también ha facilitado enormemente esta labor, sobre todo usando la especificación JPA.

Considerando los objetivos establecidos en el proyecto, indicados en el apartado 1.3, se pueden establecer las siguientes conclusiones sobre la consecucion de los objetivos descritos:

- Se ha desarrollado una aplicación web para la gestión de eventos, de grupos y de usuarios.
- Todos los datos sobre usuarios, grupos y eventos son persistidos en una base de datos.
- La interfaz es sencilla y atractiva para el usuario.
- Se ha utilizado Spring como principal framework web.
- Se ha usado el framework de Hibernate para la persistencia de datos. Además se han seguido las reglas de la especificación de JPA, trabajando con anotaciones para el mapeo objeto-relacional.
- Nos hemos acercado al concepto de Web 2.0. usando AJAX y la API de Google Maps.

Además de los objetivos inicialmente planteados, a medida que se desarrollaba la aplicación surgían nuevas ideas para mejorarla, y por ello, se han añadido nuevas funcionalidades que no estaban previstas.

Sin duda, el haber dedicado el tiempo necesario para analizar cuidadosamente los requisitos ha permitido que todos los objetivos se llevaran a cabo.

7.2. Conclusiones personales

La verdadera razón de realizar este proyecto, era la de acercarme a las nuevas tecnologías de desarrollo de aplicaciones web, y en especial a los frameworks Spring e Hibernate.

Inicialmente, mis conocimientos sobre estos frameworks eran muy escasos, y puesto que en el mundo laboral son muy utilizados, me ha parecido interesante adentrarme en el mundo de los frameworks J2EE.

Antes de centrarme en el framework que iba a utilizar, tenía 3 elecciones principales. Básicamente son los 3 frameworks más utilizados por los desarrolladores: Spring, Struts 2 o JavaServerFaces. Por lo que mi primera tarea consistió en elegir uno de los tres.

Después de buscar información por internet, me decanté por Spring. Las razones para elegir Spring fueron la gran cantidad de documentación y tutoriales que pude encontrar para iniciar el aprendizaje, algo que considero fundamental. También consideré que de los 3 frameworks anteriormente nombrados era el mejor valorado por la gente especializada.

Para la realización completa del proyecto se ha tenido que dedicar mucho tiempo en el aprendizaje de las múltiples tecnologías que se han implementado. El tiempo total invertido en esta parte, ha sido muy superior a la realización práctica del proyecto, es por eso que quiero remarcar el esfuerzo empleado en aprender, configurar y hacer funcionar estas tecnologías conjuntamente, más que en el contenido de la aplicación web,ya que ésta ha sido un medio para poder utilizarlas.

Una de las mayores desventajas conocidas de usar frameworks J2EE es que la curva de aprendizaje es muy pronunciada, algo que he podido constatar.

Si bien es cierto, que después de haber realizado este primer proyecto, considero que estos frameworks aportan muchas ventajas, y el tiempo dedicado al aprendizaje ha merecido la pena. Aunque Spring en conjunto es enorme, y sólamente he aprendido varios de los módulos más importantes, por lo que en el futuro me gustaría aprender a utilizar otros módulos como puede ser el AOP.

Por lo tanto, una vez que el proyecto está acabado, puedo decir que me ha aportado una gran cantidad de nuevos conocimientos.Y sobre todo, me ha aportado muchas ganas de seguir aprendiendo más sobre estas tecnologías.

7.3. Líneas futuras

Una de las principales motivaciones para realizar el proyecto, era aprender el framework Spring e Hibernate.

Sin embargo, las aplicaciones que se desarrollan con dichos frameworks están orientadas al mundo empresarial.

En el mundo cotidiano, las aplicaciones tienden a estar orientadas a los dispositivos portátiles. Por ello, el principal objetivo será migrar la aplicación al mundo portátil.

Además, por falta de tiempo no ha sido posible implementar algunas funcionalidades que se han ocurrido durante la realización del proyecto.

Por lo tanto, para conseguir que la aplicación sea lo más completa posible, en un futuro próximo se añadirán las siguientes funcionalidades:

- Migrar la aplicación para que pueda ser utilizada en los distintos dispositivos móviles.
- Los usuarios podrán mandarse correos entre ellos. Para poder visualizarlos cada usuario dispondrá de un buzón personal.
- Los usuarios podrán escribir mensajes en los eventos, y todos los usuarios apuntados al evento podrán leerlos.
- Se introducirán más idiomas además del español.
- Introducir más opciones para poder filtrar los registros en los listados.
- Opción de poder subir más de una foto al perfil
- Añadir un chat entre usuarios.

Glosario

AJAX	Asynchronous JavaScript And XML
AOP	Aspect-Oriented Programming
API	Application Programming Interface
BBDD	Base de Datos
CSS	Cascading Style Sheet
CRUD	Create Read Update Delete
DAO	Data Access Object
EJB	Enterprise JavaBeans
GWT	Google Web Toolkit
HQL	Hibernate Query Language
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IoC	Inversion of Control
JDBC	Java DataBase Connectivity
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JPQL	Java Persistence Query Language
J2EE	Java2 Enterprise Edition
JSON	JavaScript Object Notation
JSF	JavaServer Faces
JSP	JavaServer Pages
JSTL	JSP – Standard Tag Library
MVC	Model View Controller
ORM	Object-Relational Mapping
POJO	Plain Old Java Object
SGBD	Sistema Gestor de Base de Datos
SQL	Structured Query Language
UML	Unified Modeling Language

XML	eXtensible Markup Language

Anexo A

Planificación y presupuesto

Planificación

Debido a lo complejo que resulta el desarrollo de este proyecto, es necesario establecer una estimación inicial y planificación que defina cómo abordar las distintas fases que componen el desarrollo del mismo.

Se empieza con la parte más importante el análisis y diseño, donde se realiza la recolección de requisitos definidos en primera instancia por los usuarios, con su posterior tratamiento y desarrollo.

En segundo lugar el diseño completo del sistema, a partir de dichos requisitos dará lugar a las especificaciones necesarias para empezar el desarrollo del mismo.

A continuación, se procederá a poner en práctica el diseño obtenido, siendo fieles a lo anteriormente establecido. Esto comprenderá desde el desarrollo del código necesario hasta las posteriores remodelaciones.

Después, con las primeras versiones del sistema, se procederá a realizar los test necesarios que verifiquen el buen funcionamiento del mismo en diversas situaciones.
Debido a que en esta etapa puede surgir la necesidad de modificar el desarrollo del mismo, es posible que se deba retroceder para poder solventar los problemas y errores que puedan aparecer.

Además, se realizarán múltiples revisiones sobre la documentación creada durante todo el proceso para establecer su versión final.

Por último, se procederá a la implementación de la versión final del proyecto.

La planificación se ha establecido en horas (aproximadas) puesto que es una medida mucho más intuitiva a la hora de tomar conciencia del esfuerzo estimado.
Dicha planificación inicial se muestra en la siguiente tabla y el calendario de las tareas se representa en una diagrama Gantt.

DESCRIPCIÓN DE LA TAREA	ETAPA	HORAS ESTIMADAS
Planteamiento inicial	Análisis	12
Toma de requisitos funcionales	Análisis	8
Toma de requisitos no funcionales	Análisis	8
Reajustes de los requisitos	Análisis	8
Análisis de tecnologías candidatas	Análisis	14
Casos de uso	Análisis	4
Diagrama de casos de uso	Análisis	1
Modelo de dominio	Diseño	8
Diagrama de actividades	Diseño	8
Diagrama de flujo	Diseño	8
Diseño del sistema	Diseño	12
Diseño de las pantallas	Diseño	12
Instalación y configuración del software a utilizar	Implementación	12
Aprendizaje de las tecnologías a usar	Implementación	80
Capa dominio	Implementación	12
Capa persistencia	Implementación	12
Capa servicios	Implementación	12
Capa aplicación	Implementación	65
Capa presentación	Implementación	65
Pruebas	Pruebas	60
Documentación	Documentación	80
Despliegue	Integración	4
TOTAL		505

Tabla 5. Planificación de las tareas

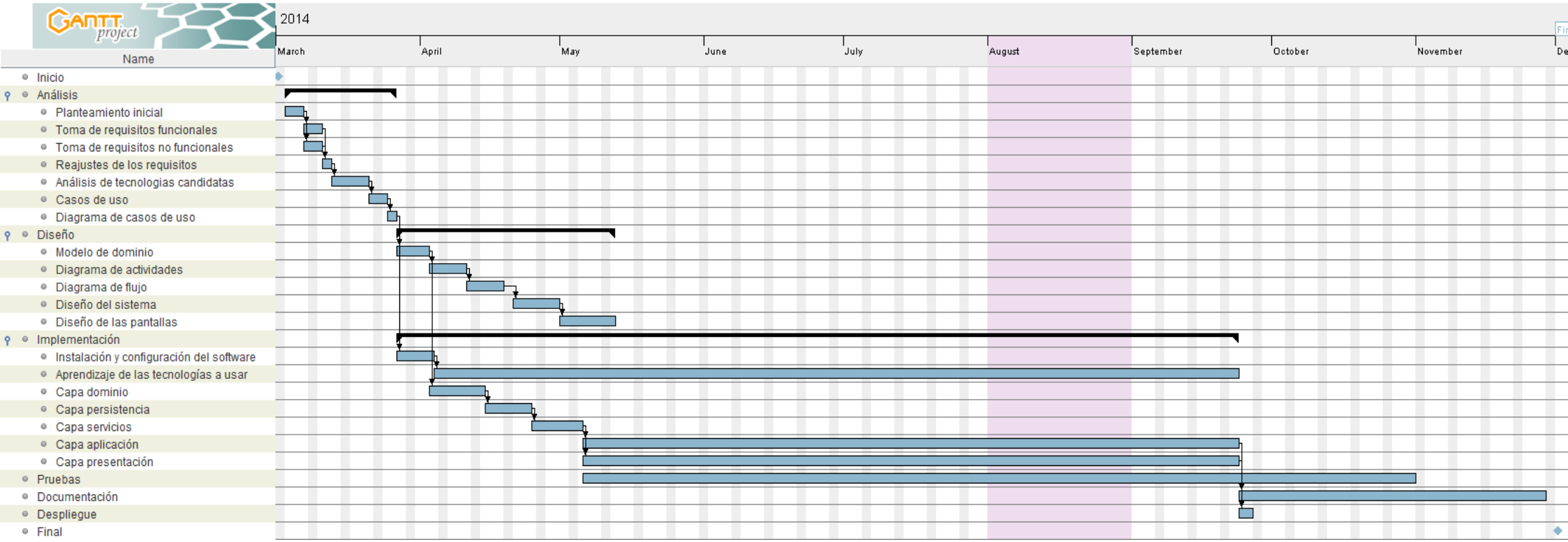

Figura 33. Diagrama de Gantt

Presupuesto

El presupuesto hace referencia al precio total real que tendría este proyecto. Se ha supuesto que se tuviese que adquirir todo lo necesario (hardware y software), aunque, claro está, cualquier empresa ya dispondría de parte de estos recursos.

Costes de personal

Éste es el coste de los recursos humanos que supondría a la empresa de desarrollo el pagar a los empleados encargados de desarrollar el proyecto según la planificación que se incluyó en la tabla 5.

Tendremos en cuenta que el precio por hora medio estándar de un analista en una empresa se paga a 36€, mientras que la del programador está a 28€. Si hacemos los cálculos pertinentes:

Recurso	Número	Importe/hora	Nº Horas	Importe total
Analista	1	36€	103	3708€
Programador	1	28€	402	11256€
Total	2	-	505	14964€

Tabla 6: Costes de personal

Costes de material para el desarrollo

El coste de los recursos informáticos para el desarrollo, pueden convertirse en uno de los más significativos dentro de un proyecto y a la vez de lo más difíciles a la hora de calcular. Los costes de los productos son caros, pero además no se pueden atribuir exclusivamente a un proyecto concreto puesto que, generalmente, lo podemos emplear para posteriores proyectos. Una de las cualidades de nuestro proyecto, es el uso de tecnologías Open Source, eso nos reduce los gatos exclusivamente a temas de hardware.

Descripción	Coste	% Uso	Dedicación (meses)	Período de depreciación	Coste imputable
Servidor	3000€	100%	7	60 meses	452€
Ordenador	1500€	100%	8	60 meses	257€
S.O. Windows 8	0,00€	100%	8	-	0,00€
Base de datos MySql	0,00€	100%	7	-	0,00€
Servidor de aplicaciones Tomcat	0,00€	100%	7	-	0,00€
Framework Hibernate	0,00€	100%	7	-	0,00€
Framework Spring 3 MVC	0,00€	100%	7	-	0,00€
Eclipse IDE	0,00€	100%	7	-	0,00€
TOTAL					709€

Tabla 7: Costes material para el desarrollo

Costes de subcontratación de tareas

Éste sería el coste derivado de contratar una empresa de software para el desarrollo de este producto. En este caso, no hay ningún gasto en este concepto puesto que el producto ha sido desarrollado por profesionales de la propia empresa.

Otros costes

No se ha contemplado ningún otro tipo de coste como pueden ser viajes o dietas.

Resumen de costes

Concepto	Presupuesto
Personal	14964€
Material para el desarrollo	709€
Subcontratación de tareas	0€
Otros costes	0€
TOTAL sin I.V.A	15673€
TOTAL con I.V.A	18964,33€

Tabla 8: Resumen de costes

El presupuesto total del proyecto sin I.V.A asciende a la cantidad de 15673 EUROS y aplicando el 21% del I.V.A correspondiente, el precio final del proyecto es de 18964,33 EUROS.

Anexo B

Comparativas de frameworks J2EE

Los Web frameworks son todos muy diferentes y se han creado normalmente por diferentes razones y para lograr diferentes objetivos. Hay muchas características que pueden influir en la decisión de elegir uno u otro y, por supuesto, dependerá del tipo de aplicación que se está construyendo.

En este anexo vamos a analizar las ventajas y desventajas de los frameworks Web más populares del mercado.

En primer lugar, vamos a mostrar la siguiente comparativa para ver qué cuota de mercado tiene cada framework.

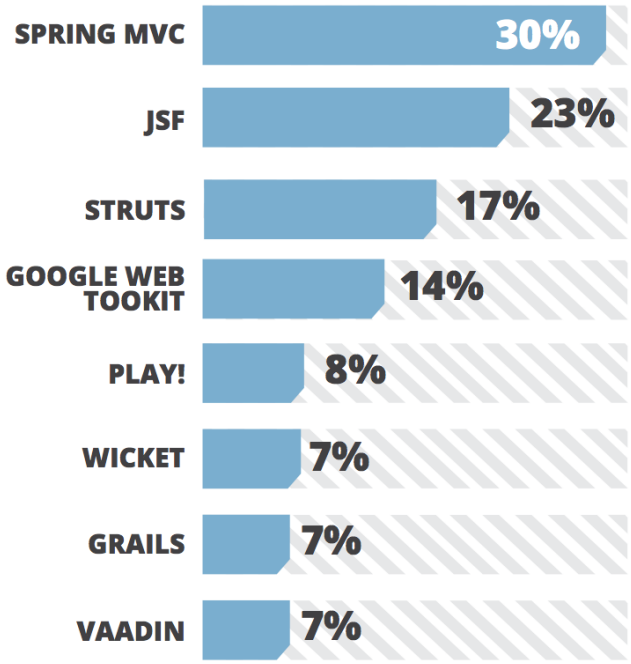

Figura 34. Cuota de mercado en 2012 de los principales frameworks [23]

Como podemos ver, aunque JSF es el estándar, Spring MVC hoy por hoy es el framework web más utilizado.

Veamos ventajas y desventajas de los más importantes.

Spring MVC 3

Ventajas

- Integración con diferentes opciones para la vista como JSP/JSTL, Tiles, Velocity, FreeMarker, Excel, PDF, o implementar tu propio lenguaje para integrarlo en la vista de la aplicación.
- Los controladores de Spring MVC se configuran mediante IoC como los demás objetos, lo cual los hace fácilmente testeables e integrables con otros objetos que estén en el contexto de Spring, y por tanto sean manejables por éste.
- Spring soporta un gran número de especificaciones, como JSR330, JMS, JSR250, JPA, JSR303, etc…
- En cuanto a salida laboral, es muy conocido.
- Al ser el framework más utilizado, existe mucha documentación y tutoriales disponibles en Internet.
- Constantemente actualizándose

Desventajas

- El contenedor de Spring no es ligero (si se usan todos los módulos disponibles), no es recomendable su uso en aplicaciones de tiempo real o en aplicaciones para móviles.
- No se puede evaluar si un objeto ha sido bien inyectado más que en tiempo de ejecución. Aunque hay herramientas como Spring IDE que sí que ayudan.
- Spring es bastante complejo y extenso, es necesario un tiempo previo para poder explotar todas sus cualidades.

Struts 2

Ventajas

- Arquitectura simple y fácil de extender.
- Framework especializado en controlador.
- Librerías de Tags fáciles de personalizar.
- AJAX integrado mediante Dojo Tookit.
- Plugins para integrar GWT y resultados JSON.
- OGNL ayuda en la conversión y validación en la parte del cliente.
- Facilidad en la testeabilidad.
- Soporta la internacionalización y apuesta por la separación de ficheros por cada página y acción en la internacionalización.
- Puede usar Tiles & SiteMesh para la decoración de páginas.

Desventajas

- Documentación mal organizada.
- Struts 1.x abarca la mayoría de resultados en las búsquedas de internet.

- Muchas clases y archivos de configuración deben ser escritos para un solo componente.
- Curva de aprendizaje muy marcada.

Java Server Faces

Ventajas

- JSF forma parte del estándar J2EE
- Fácil de usar y no hay dependencias externas, siempre y cuando permanezca dentro del ecosistema de Java EE, que JSF aprovecha bien.
- Es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.
- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n).
- Puede usar Tiles & SiteMesh para la decoración de páginas.
- JSF ofrece una gran cantidad de componentes open source para las funcionalidades que se necesiten. Los componentes Tomahawk de MyFaces y ADFFaces de Oracle son un ejemplo. Además, también existe una gran cantidad de herramientas para el desarrollo IDE en JSF al ser el estándar de JAVA.

Desventajas

- Su naturaleza como estándar hace que la evolución de JSF no sea tan rápida como pueda ser la de otros entornos como WebWork, Wicket, Spring, etc.
- No trabaja bien con REST.
- Se requiere de varias fuentes para la implementación.
- No soporta AJAX, se tiene que recurrir a librerías como Ajax4JSF.
- JSF es increíblemente complejo debido a la especificación Java EE.

Tapestry

Ventajas

- Muy productivo una vez se aprende como funciona.
- Muchas actualizaciones e innovaciones entre versiones.
- Integra AJAX mediante Dojo Toolkit
- Buen sistema de validación.
- Soporta la internacionalización y apuesta por la separación de ficheros por cada página y acción en la internacionalización.
- Puede usar Tiles & SiteMesh para la decoración de páginas.

Desventajas

- Mala documentación.

- Curva de aprendizaje lenta.
- El tiempo entre actualizaciones es muy largo.

Google Web Toolkit

Ventajas

- Permite a los desarrolladores crear aplicaciones Javascript con interfaces complejas a partir de código Java.
- El compilador de GWT se encarga de optimizar las clases Java para crear el mínimo de Javascript necesario para correr nuestra aplicación.
- El compilador de GWT se va a encargar de crear código Javascript optimizado para los distintos navegadores.

Desventajas

- Curva de aprendizaje lenta al principio si el programador tiene mucha experiencia en otra tecnología
- El GWT, tiene como dificultad el funcionamiento correcto en algunos IDE's, que no sean Eclipse, como Netbeans por ejemplo.
- Como el compilador de GWT traduce el lenguaje de las clases a HTML y demás, el tiempo de compilación es muy lento, lo que hace que es un poco ineficiente en rapidez

Anexo C

Patrón DAO

En una aplicación J2EE necesitamos acceder a datos, ya sea por persistencia (hibernate, jdo, iBatis…), jdbc, ficheros de texto, LDAP, etc…. Esto puede suponer un problema, pues la forma de acceder a los datos depende del fabricante y del tipo de almacenamiento que estamos accediendo.

Los componentes de nuestra aplicación deben ser transparentes en la medida de lo posible al actual sistema de persistencia o fuente de datos para permitir migraciones entre distintos fabricantes, distintos tipos de almacenamiento y diferentes fuentes de datos. Supongamos que en un momento dado queremos cambiar el motor de persistencia. Siguiendo este módelo será mucho más fácil.

Debemos tener muy claro que el acceso a los datos varía mucho dependiendo de la fuente de los datos. El acceso al almacenamiento persistente, como una base de datos, varía mucho en función del tipo de almacenamiento (bases de datos relacionales, bases de datos orientadas a objetos, archivos planos, etc) y la implementación de cada uno de los proveedores que tiene la empresa.

El patrón DAO viene a resolver este problema usando un Objeto de Acceso a Datos para abstraer y encapsular el acceso a los datos. Un DAO maneja la conexión con la fuente de datos para obtener y guardar los datos. De esta forma conseguimos que nuestra lógica de negocio no sepa nada del motor de persistencia, y siempre que quiera acceder a los datos lo hará usando la interface DAO, con esto conseguimos reducir el acoplamiento. [24]

Un DAO siempre realiza operaciones atómicas contra la base de datos, nunca son necesarias las transacciones. Claros ejemplos de esto serían busquedas por una clave, creación, actualización y borrado de un registro, obtener todos los registros y cualquier otra operación que vayamos a realizar muy a menudo.

Normalmente se crea un DAO por cada Objeto que usemos en nuestra aplicación, como por ejemplo un Entity de Hibernate.

Los componentes de negocio que se basan en DAO utilizan la interfaz más simple expuesta por DAO para sus clientes. DAO oculta completamente los detalles del origen de datos de la aplicación hacia sus clientes. Debido a que la interfaz expuesta por el DAO a los clientes no cambia cuando los datos subyacentes cambian su implementación de código, este modelo permite al DAO adaptarse a los sitemas de almacenamiento sin que ello afecte a sus clientes o componentes de negocio.

BusinessObject

El BusinessObject es la clase que va a obtener y almacenar los datos que obtendremos con el patrón DAO, por lo tanto requiere el acceso a la fuente de datos .

DataAccessObject

Es el objeto principal de este patrón. El DataAccessObject abstrae la implementación del acceso a los datos subyacentes para el BusinessObject para permitir el acceso transparente a la fuente de datos. El BusinessObject también delega la carga de datos y operaciones de almacenamiento al DataAccessObject.

El DAO provee una interface permitiendo operaciones específicas sin exponer los detalles de cómo se accede a la base de datos. Una buena práctica,es que estas interfaces incorporen la funcionalidad CRUD.

DataSource

Contiene la implementación específica para cada fuente de datos.

DTO (Data Transfer Object)

Son objetos para transferir datos, su función es pasar los datos de una capa a otra.
Proporcionan una representación de otro objeto, o pueden ofrecer una representación de elementos de datos de varios objetos, que son por lo general, objetos relacionados (dependencias)
Pueden o no ofrecer todas las propiedades del objeto que representan

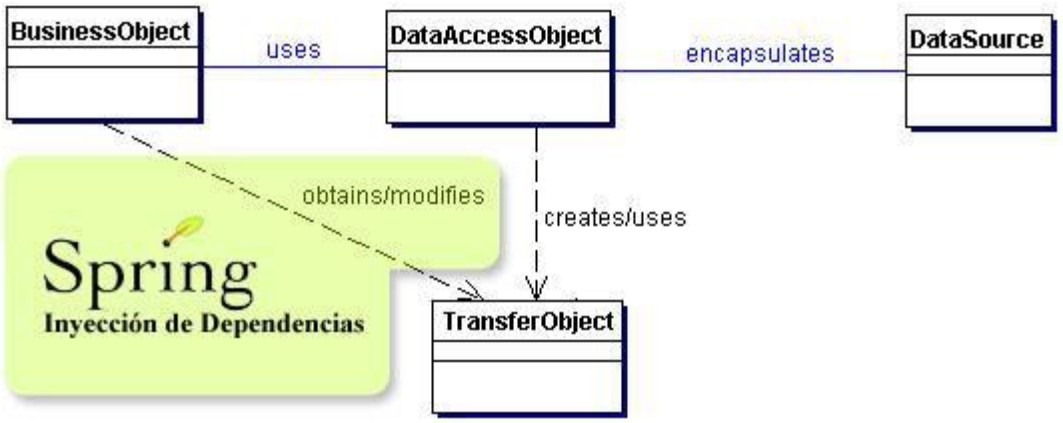

Figura 35. Esquema de patrón DAO

Ventajas

- Los DAO son un patrón de diseño J2EE y es considerada como buena práctica.
- Los objetos de negocio no conocen como se va a manipular la información, ellos sólo tienen acceso a unos servicios para almacenar o obtener información de la base de datos.
- Separación entre la capa de persistencia y la aplicación.
- Los cambios realizados en la capa de persistencia no afectan a los clientes que usan DAO.

Desventajas

- Al agregar un DAO, la complejidad de usar otra capa aumenta la cantidad de código ejecutado durante el tiempo de ejecución.

Anexo D

Patrón MVC

El patrón MVC fue originalmente formulado a finales de los 70s por Trygve Reenskaug en Xeros SPARC como parte del sistema Smalltalk. [25]

Este patrón está indicado especialmente para el diseño de arquitecturas de aplicaciones que requieran de una gran interactividad con los usuarios, como es el caso de aplicaciones Web. De hecho, la gran mayoría de Frameworks web J2EE están basados en el patrón MVC.

Es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.

El principio fundamental del patrón MVC es definir una arquitectura con claras responsabilidades para diferenciar componentes.

Por un lado tenemos el Modelo, el cual representa los datos de la aplicación y sus reglas de negocio; por otro la Vista, compuesta de vistas que representan los formularios de entrada y salida de datos; y finalmente, el Controlador, encargado de procesar las peticiones entrantes del usuario y controlar el flujo de ejecución del sistema.

El patrón MVC en la programación web J2EE se le conoce como arquitectura de modelo 2. Esta arquitectura consiste en la utilización de Servlets para procesar las peticiones, que estarían contenidos en el Controlador del patrón, y páginas JSP para mostrar la interfaz del usuario que representaría la Vista, y finalmente los famosos JavaBeans ubicados en el modelo.

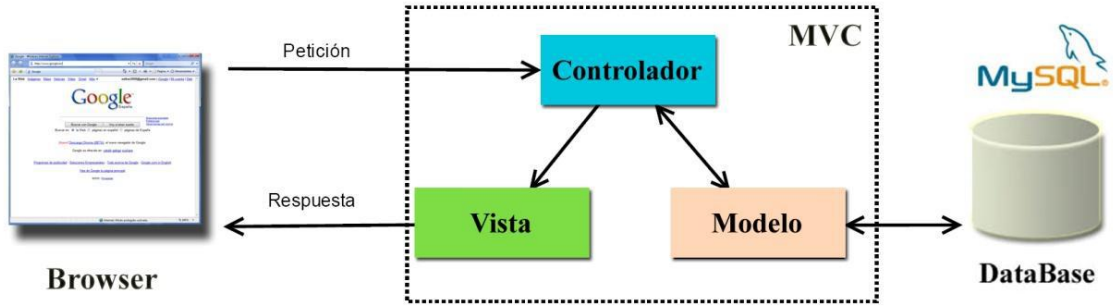

Figura 36. Esquema de Patrón MVC

Controlador
Todas las peticiones a la capa intermedia que se realicen desde el cliente pasarán por el Controlador, éste determinará las acciones a realizar e invocar al resto de los componentes de la aplicación como pueden ser el modelo o la vista.

Vista
La vista es la encargada de generar las respuestas que deben ser enviadas al cliente. Esta respuesta normalmente incluirá datos generados por el controlador, entonces el contenido de la página no será estático sino que será generado de forma dinámica, y ahí es donde entrarán los JSP.

Modelo
Encapsula la lógica de negocio de la aplicación, acceso a los datos y su manipulación.

Debido al auge que tienen las aplicaciones web basadas en Ajax, la aplicación del patrón MVC ha sido mejorada para proveer mejores experiencias a los usuarios usando JavaScript, Ajax y el uso de formatos especiales de información como son JSON o XML.

Anexo E

Guía de usuario

Este anexo sirve como tutorial para guiar al usuario a través de la aplicación “Woodie”.

Debido a que la aplicación gestiona usuarios, regiones, eventos y grupos se dividirá la guía en 4 secciones.

Sección 1. Gestión de usuarios

Tanto si se está registrado como si no, el usuario siempre accederá a la aplicación por medio de la pantalla de inicio. En ella, el usuario accede a la pantalla de login si ya está registrado, o bien a la pantalla de registro, si no está registrado en el sistema.

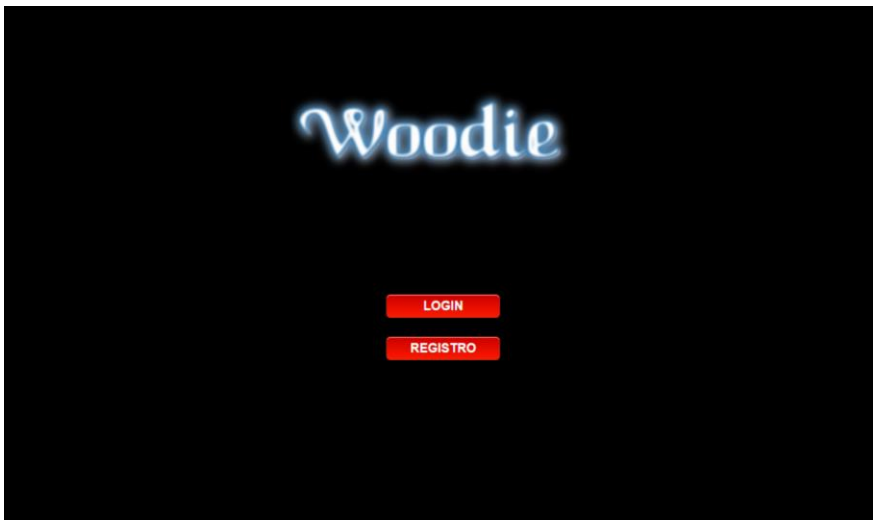

Figura 37. Pantalla de inicio

En la pantalla de autenticación el usuario tiene que introducir su nombre de usuario y su contraseña. En caso de que no sean correctas, el sistema indica un mensaje de error indicando que los datos introducidos no son correctos.

Figura 38. Pantalla de login

En el caso de acceder a la pantalla de registro de usuario, éste tendrá que introducir sus datos personales. Campos obligatorios serán el nombre de usuario y el password. También el usuario podrá asociar una imagen al perfil.

Figura 39. Pantalla de registro.

Una vez que el usuario accede a la aplicación, ésta mostrará siempre un menú lateral con las distintas secciones y una cabecera con el nombre del usuario y el botón para salir de la aplicación.

Si se pulsa en el nombre del usuario, se accederá directamente al perfil del usuario.

Si se pulsa en el botón del menú “Usuarios” se accederá al listado de usuarios, donde podemos realizar búsquedas de un usuario en particular.

Sección 2.Gestión de eventos.

Si se pulsa el botón del menú lateral “Eventos” se accederá al menú de los eventos, donde podemos elegir entre crear un evento, o mostrar algún tipo de listado.

Para crear un evento, hay que introducir algún dato obligatorio como es el nombre del evento. También se pueden indicar lugar, fecha, hora, descripción, etc.

Hay un mapa de Google Maps disponible para facilitar la localización y un buscador de geolocalización.

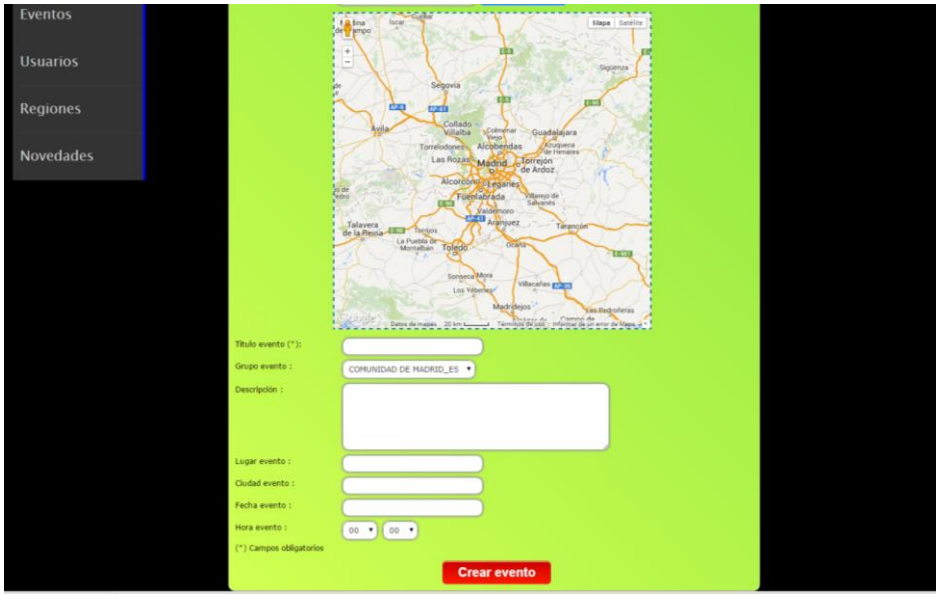

Figura 40. Pantalla de crear Evento

En cuanto a los distintos tipos de listados, el usuario podrá realizar búsquedas por nombre de evento. Los eventos se muestran ordenados por fecha de evento.

Al lado de cada registro, se muestran varios posibles enlaces como pueden ser ver la información del evento, modificar el evento, borrar el evento, apuntarse al evento y abandonar el evento.

Sección 3.Gestión de grupos

Si se pulsa el botón del menú lateral “Grupos” se accederá al menú de los grupos, donde podemos elegir entre crear un grupo, o mostrar un listado.

Para crear un grupo, hay que introducir algún dato obligatorio como es el nombre del grupo. También se pueden añadir usuarios al grupo o eliminar usuario del grupo.

En cuanto al listado, el usuario podrá realizar búsquedas por nombre de grupo.

Al lado de cada registro, se muestran varios posibles enlaces como pueden ser ver la información del grupo, modificar el grupo, borrar el grupo, apuntarse al grupo y abandonar el grupo.

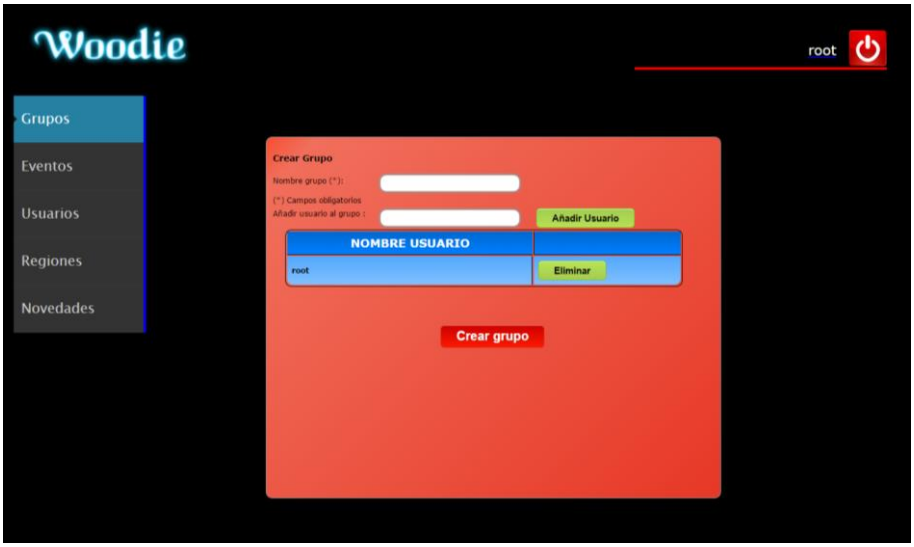

Figura 41. Pantalla de crear grupo

Sección 4. Gestión de regiones

Sólo el usuario administrador puede gestionar las regiones, por lo que es el único que tendrá a disposición el botón del menú lateral “Regiones”.

Al pulsar el botón se muestra el listado de las regiones registradas en el sistema, pudiendo modificar o borrar las regiones del listado. También se pueden realizar búsquedas por nombre de región. Y se muestra un botón para poder crear nuevas regiones.

Para crear una región hay que indicar el nombre de la región y el nombre del país al que pertence, ambos son campos obligatorios.

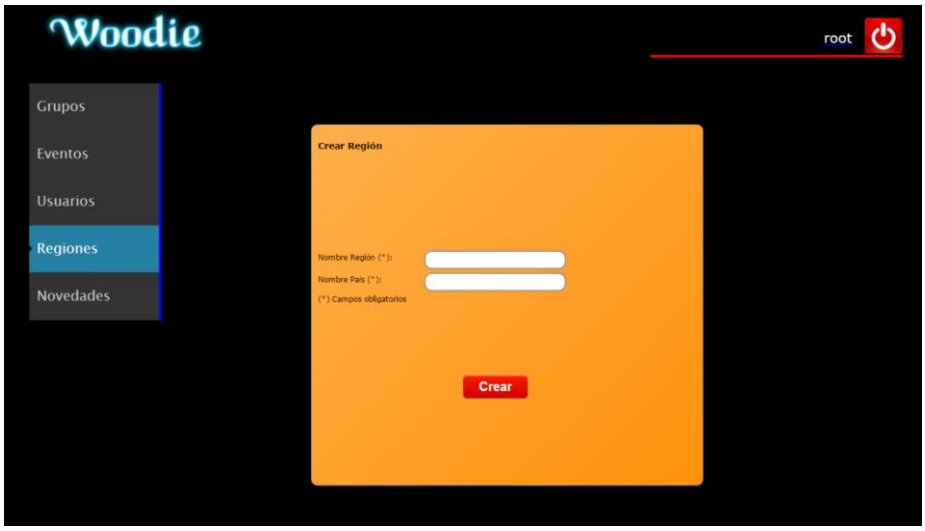

Figura 42. Pantalla de crear región

Anexo F

Material entregado

Para poder desplegar correctamente la aplicación Woodie en el servidor, se han de cumplir una serie de requisitos previos:

- Tener instalada, la máquina virtual java (versión JDK 7).
- Tener instalado el servidor Apache Tomcat (versión 7.0).
- Tener instalado MySQL.
- Haber creado una base de datos llamada “woodie” en MySQL.

Todos los programas necesarios para poder arrancar la aplicación se entregan con el material adjunto a la memoria del proyecto.

Los ficheros incluidos para instalar y establecer las condiciones iniciales son entregados en una jerarquía de directorios. Los directorios son los siguientes:

- Base de datos: en este directorio se incluyen el instalador de MySql y un script para insertar los datos inciales en las tablas.
- Servidor: en este directorio se incluyen el instalador de la máquina virtual de Java, y el instalador del servidor Tomcat 7.
- Woodie: en este directorio se incluye el fichero woodie.war, que es el fichero que contiene el proyecto listo para ser desplegado en el contenedor de Servlets Apache Tomcat.

Finalmente, sólo tenemos que copiar el fichero woodie.war dentro de nuestro Contenedor de Servlets Tomcat en la carpeta webapps:
C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps

Iniciamos Apache Tomcat y desde el browser escribimos: http://localhost:8080/woodie

Bibliografía

[1] Walls, Craig. Spring in Action 3 .2011

[2] Acera García, Migue Angel. CSS 3.2012

[3] Pugh, Eric. Professional Hibernate. 2004

[4] La Web 2.0. Una revolución social y creativa. 2009

Disponible:
http://telos.fundaciontelefonica.com/telos/articulodocumento.asp@idarticulo=3&rev=74.htm

[5] Análisis de los servicios de la tecnología Web 2.0 aplicados a la educación. 2010
Disponible:
http://www.nosolousabilidad.com/articulos/tecnologia_educacion.htm#sthash.S2qffYht.dpuf

[6] Qué es AJAX. 2012

Disponible:

http://www.digitallearning.es/blog/que-es-ajax/

[7] JSON. 2011

Disponible:

http://librosweb.es/symfony/capitulo_11/json.html

[8] Qué es CSS. 2012

Disponible:

http://librosweb.es/css/capitulo_1.html

[9] Wikipedia. Google Maps. 2014

Disponible: http://es.wikipedia.org/wiki/Google_Maps

[10] Introducción a Spring 3. 2010

Disponible: http://www.javatutoriales.com/2010/09/spring-parte-1-introduccion.html

[11] Spring: Introducción + inyección de dependencias. 2013

Disponible: http://todosobreprogramacion.blogspot.com.es/2013/09/spring-introduccion.html

[12] Spring Framework, módulos y críticas. 2014

Disponible: http://centrodeartigo.com/articulos-informativos/article_69801.html

[13] Programación Orientada a Aspectos (AOP , Aspect-Oriented Programming)

Disponible: http://nullbrainexception.blogspot.com.es/2009/11/programacion-orientada-aspectos-aop.html

[14] Spring Framework Reference Documentation. 2014

Disponible: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/index.html

[15] Spring MVC Framework Tutorial. 2012

Disponible: http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[16] Spring MVC Configuración. 2013

Disponible: http://www.arquitecturajava.com/spring-mvc-configuracion/

[17] Hibernate: Framework para el mapeo objeto-relacional. 2010

Disponible: http://carlos-henriquez.blogspot.com.es/2010/07/hibernate-framework-para-el-mapeo-de.html

[18] Tutorial de Hibernate. 2010

Disponible: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate

[19] Documentación de referencia de Hibernate. 2014

Disponible:https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html/index.html

[20] Interfaz de Persistencia Java (JPA) - Entidades y Managers.

Disponible: http://www.lab.inf.uc3m.es/~a0080802/RAI/jpa.html

[21] Qué es JPA. 2010

Disponible: http://www.aprendiendojava.com.ar/index.php?topic=54.0

[22] Introduccion a JPA (Java Persistence API). 2008

Disponible: https://ubuntulife.wordpress.com/2008/10/20/introduccion-a-jpa-java-persistence-api/

[23] The Curious Coder's Java Web Frameworks Comparison: Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts and JSF. 2013

Disponible: http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/

[24] Spring Framework: El patrón DAO. 2011

Disponible: http://www.genbetadev.com/java-j2ee/spring-framework-el-patrn-dao

[25] Introducción al patrón Modelo-Vista-Controlador con Spring MVC. 2011

Disponible: http://www.e-continua.com.mx/index.php/15-springmvc